MACRO-ASSEMBLER SNAP/3

User's Guide

Version 3

April, 1980

Document No. 50296

DATAPOINT

MACRO-ASSEMBLER
SNAP/3

User's Guide

Version 3

April, 1980

Document No. 50296

PREFACE

The SNAP/3 assembler runs on any Datapoint processor with at least the 5500 instruction set and can assemble programs for any Datapoint processor. SNAP/3 contains all of the features of SNAP/1 and SNAP/2 but runs much faster, especially when assembling programs with many macros. SNAP/3 also assembles the additional instructions accepted by the Datapoint 6600 processor. SNAP/3 can produce either an absolute object program file or a relocatable object file; a relocatable file must be processed by the LINK utility before it can be executed.

TABLE OF CONTENTS

CHAPTER 1. INTRODUCTION

1.1 Changes to SNAP/3 since version 1

1. Two new options, "B" and "H", were added to the command line and the SNAPOPT directive to allow numbers in the program listing to be edited in binary or hexadecimal instead of octal. See sections 3.19 and 6.1 for more details.

1.2 Changes to SNAP/3 since version 2

1. SNAP/3 now supports text file libraries. Both source files and include files may now be assembled directly from a text library. Libraries are created and manipulated using the LIBRARY command. A member name is specified for a file by placing a period (.) after the normal DOS file specification, and then the library member name. For example:

```
INCLIB.DEFINE
MAINLIB:DR4.INCLUDE
```

   Also note that if a member name is given for a file, the default extension becomes "LIB" instead of "TXT". On the command line, both the source file and include (5th file spec) may specify library members. (See the LIBRARY user's guide for more information).

2. Hexadecimal and Binary constants may now be specified in the expression field. A hex constant is preceeded by an ampersand (&), and a binary constant is preceeded by a percent sign (%).

3. For users of the ARC system, the time and date now appears on the listing. If a valid time can be found from file ARCCLOCK/TXT (See the ARC user's guide for more information), it will be printed just below the user heading on the front page, and on cross-reference pages. Note that the time is updated between cross-references and PROGs.

4. If the "P" or "Q" option is given without the "L" option, the program name, program address blocks, and transfer address

will be printed on the listing.

5. Many more inclusions are now possible. After inclusion "Z", the next inclusion will be "a". After "z" will be "0". After "9", more inclusions are possible, but the inclusion letter will be undefined.

6. The "?" option has been added. This causes the command line format and options to be displayed. No assembly will be performed if the option is given.

7. The INFO instruction no longer requires the "6" option. This instruction is now defined to be a 5500 instruction.

1.3 Introduction to SNAP/3

SNAP/3 may be used to generate either absolute or relocatable object code from a source program file. The file may be created using the EDITOR, and consists of mnemonic instructions, assembler directives, and comments.

The kind of object file produced is controlled by a command-line option. An absolute object file may be loaded for execution by the DOS loader, while a relocatable object program must be processed by the LINK utility to create an absolute program.

Since SNAP/3 and this manual assume many details which are inherent to the DOS and Datapoint processors, a working knowledge of both the DOS and processor is recommended before proceeding.

Basically, the SNAP/3 assembler is a program that assigns numerical values to symbols and puts out these values upon input of the associated symbols. Symbols in certain fields have preassigned values (such as instruction mnemonics) while other symbols are defined by the user (such as labels and macro names).

The value assigned to an instruction mnemonic is the binary bit configuration recognized by the processor for that instruction. For example, the following instruction mnemonics have the following octal values:

MNEMONIC	VALUE
ADBC	0062 0201
RET	0007
SU	0024

Predefined symbols are kept separately by SNAP/3 and recognized as reserved symbols only when they are encountered in the proper context. In context other than that where their usage is predefined, the symbol will assume whatever value the user may wish to assign. For example:

LABEL	INSTRUCTION	EXPRESSION
L1	AD	1
	JMP	CALL
L2	AD	2
CALL	CALL	SUBR1
INPUT	INPUT	

There is no problem in differentiating the two CALL and INPUT symbols since the ones in the instruction field are predefined and the ones in the label and expression fields are user-defined.

Along with relating symbols to numbers, another major function of the SNAP/3 assembler is to enable the programmer to reference a symbol that is defined later in the program. This is called FORWARD REFERENCING, and may be handled in a variety of ways. When SNAP/3 is generating relocatable output, the forward references are resolved by the LINK utility using information in the relocatable file. When SNAP/3 is generating absolute output or a code listing is requested, it produces an intermediate internal file similar to a relocatable output file which it reads back during a second "pass" and produces the actual relocatable or absolute output file and/or the listing with the resolved forward references. A second pass may also be requested by an option on the command line; this option is necessary if the relocatable output file is to be loaded by the DOS relocatable loader function (function 15).

An optional function of SNAP/3 is that of producing a tabulated listing of all user-defined symbols, their octal value, and all references to them. This cross-reference table generation consists of recording all definitions of and references to user-defined symbols, sorting the references, and merging them with their values.

SNAP/3 maintains two internal counters called the ADDRESS COUNTER and the LOCATION COUNTER. The ADDRESS COUNTER indicates the memory address of the object code currently being generated and the LOCATION COUNTER indicates the memory address at which the object code currently being generated will be executed. Thus it is possible to assemble code which may be loaded into memory at any address, but which will execute properly only when loaded at

the address given by the LOCATION COUNTER. These counters are usually the same except in the case of Located Code which is generated by the LOC directive (see section 3.10). Each time a byte of code is generated, both counters are incremented. The values of these counters are initially set to 010000 but directives are available for changing their values either initially or dynamically (see sections 3 and 5). The content of the LOCATION COUNTER when processing of the current line is initiated is usually displayed at the left side of the listing. The dollar sign character ($) has special meaning in that it has the value of the LOCATION COUNTER when processing of the current line began. For example:

```
ADRCTR     OBJECT CODE     SOURCE CODE

01000                              SET   01000
01000      104 000 002     XXX     JMP   XXX
01003      104 003 002     DOG     JMP   $
01006                      A       EQU   $
00001                      B       EQU   1
01006      123 123                 DC    0123,83
05400L                             LOC   05400
05403L                     C       EQU   $+3
```

SNAP/3 maintains a stack of 15 dynamic Program Address Blocks (PAB's) which may be used to locate data and code at Assembly time. A PAB is actually an ADDRESS COUNTER which has been given a symbolic name. This name is not used as a dictionary entry but is used solely for the purpose of requesting an ADDRESS COUNTER swap with the current PAB (see sections 3.14 and 3.24).

An ABSOLUTE PAB is defined by SNAP/3 and is implicitly used anytime the programmer neglects to Originate (ORG) and Use (USE) additional PAB's (see section 3.14 and 3.24). When a new PAB is requested, the current PAB's ADDRESS COUNTER is stored and the next available address associated with the requested PAB is placed in the ADDRESS and LOCATION COUNTERS.

The first word address and the length of each PAB is printed at the end of pass one.

Example of PAB usage:

```
ADRCTR  OBJECT CODE     SOURCE CODE

01000                   BUFFER  ORG   01000
07000                   CODE    ORG   07000
00120                   LTH     EQU   80
```

```
07000                                   USE     CODE
07000    002 000 120                    DC      *BUF1,LTH
07003    002 120 120                    DC      *BUF2,LTH
01000                                   USE     BUFFER
01000                           BUF1    SK      LTH
01120                           BUF2    SK      LTH
07006                                   USE     *
07006    377                            HALT
```

Object code generated by SNAP/3 will be assumed to be non-relocatable starting at octal location 010000 until an "ORG location-zero" directive is given followed by a USE statement referencing the ORG 0 program address block. A non-zero origin for any program address block (PAB) will render the generated object code for that address block non-relocatable.

A description of the format of an absolute object file may be found in the DOS User's guide. A description of the format of a relocatable object file may be found in the LINK User's Guide.

CHAPTER 2. STATEMENTS

An assembly code statement consists of a label field, an instruction field, an expression field and a comment field. For example:

```
  1        2        3                4
LABEL     JTC     START    THIS IS A COMMENT FIELD
```

Field 1 is the LABEL FIELD
Field 2 is the INSTRUCTION FIELD
Field 3 is the EXPRESSION FIELD
Field 4 is the COMMENT FIELD

The editor provides tabulation so that the fields may be justified to begin in a certain column for ease of reading. Tab stops at columns 11, 21 and 38 create a good appearance. However, SNAP/3 only requires the following:

A non-space in the first column means that the first field is a label except for a leading period (.), plus (+), or asterisk (*), which designate the entire line as a comment line.

Instruction mnemonics, SNAP/3 directives and SNAP/3 macro names must start at or before column 20.

Expressions must start at or before column 25.

Any statements which are blank prior to column 21 will be treated as comments.

Scanning proceeds from left to right with one or more spaces serving as field delimiters.

2.1 Label field

The label field may consist of from one to eight characters. If more than eight are used only the first seven and last will be used as a label name in the dictionary and therefore, must be unique. The first character may be any alphabetic character or a dollar sign ($). The other characters may be any alphanumeric character or a dollar sign. A terminating asterisk (*) will

declare the label as a fixed program entry point and the label will be written to an entry point file by SNAP/3 (see section 6.1). If the label is terminated by a colon (:), the label will be declared an external definition to be used by the linkage editor in resolving external references. If SNAP/3 is producing an absolute object file, a label terminated by a colon will be treated like a label terminated by an asterisk, as a fixed entry point. If the label is terminated by an equal sign (=), and the label has been previously defined, a redefinition of the label's value will occur and the normal "D" error flag will not be generated. Extreme care must be exercised when using this redefinition capability as any reference to a multiply defined label will use the most recently defined value, or the last definition if the label has not been previously defined. Note, however, that the colon, asterisk, or equal sign is not part of the label itself. Thus when the label is referenced in the operand, only the name, without the designator, is used. Some examples of labels follow.

VALID LABELS

```
LBL12
LABEL$
LABELA*
LABELB:
LABELC=
```

INVALID LABELS

1LABEL	Starts with numeric.
LABEL#	Non-alphanumeric or $ character (#).
LABEL.	Non-alphanumeric or $ character (.).
L1-2L3	Non-alphanumeric or $ character (-).

Invalid labels will be flagged with an "E" error flag.

The following characters have special meaning when they appear in column one:

. A period in the first column will cause SNAP/3 to treat the entire line as a comment line.

+ A plus sign in the first column will cause a page eject during the listing of the program. The line will be

treated as a comment line as well and printing will occur after the ejection.

* An asterisk in the first column will cause a page eject if the listing is within two inches of the bottom of a page. The line is treated as a comment line and printing occurs after any possible ejection.

2.2 Instruction field

The instruction field may be any of the instruction mnemonics, SNAP/3 directives, or a macro name. It has the same syntactical restrictions as the label field (up to eight characters starting with a letter or dollar sign ($) and containing only alphanumerics or dollar signs).

Only the following instruction mnemonics and SNAP/3 directives may be abbreviated.

INPUT abbreviated as IN
JUMP abbreviated as JMP
LIST abbreviated as LIS
RETURN abbreviated as RET
SKIP abbreviated as SK

Any illegal or undefined instruction mnemonics will cause "I" error flags to be generated.

2.3 Expression field

The expression field consists of one or more expressions, delimited by commas (,), comprising any number of strings, numbers, or symbols with operators between them. Supplying more expressions than are permitted for an instruction or directive will result in in "E" error flag. A space after an operand or right parenthesis terminates the expression and expression field. Spaces are ignored after a left parenthesis or operator.

Numbers are assumed to be decimal (base 10) unless they start with a special character. If the number is octal (base 8), it must contain at least one leading zero. If the constant is to be taken as hexadecimal (base 16), it must begin with an ampersand (&). Binary (base 2) numbers begin with a percent (%) sign. 12 is 12 decimal, 023 is 023 octal (19 decimal), &F is the hex number "F" (17 decimal), and %010110 is the decimal number 22. String quantities are denoted (preceded and followed) by apostrophes (').

The DC directive allows strings containing one or more characters. All other directives and instructions allow strings of only one character in length. The numeric value of a character is its ASCII binary value with the parity bit always a zero. A null string is illegal. A forcing character (#) is used in strings to indicate that the next character should be taken as ASCII no matter what it is. This is useful for entering the characters (') and (#) themselves into the string. For example:

'#'##' is the character string '#

Expressions are evaluated from left to right and all operators have the same precedence. The order of evaluation may be modified with the use of parentheses as in arithmetic expressions. For example, the following is a legitimate expression in SNAP/3 :

(ADDRESS1<8)-ADDRESS2/8+(ADDRESS3-ADDRESS4)

The expression scanner generates a 16-bit two's complement value giving a range of -32768 through +32767. Instructions which use only eight bits will discard the most significant byte (MSB) of the value generated by the expression scanner and use ony the least significant byte (LSB) of the value. Syntax errors in expressions will be flagged with "E" error flags.

Undefined labels in the expression field of DA, DC, and statements containing instruction mnemonics will be treated as external references to be resolved by the linkage editor if SNAP/3 is producing relocatable output. The statements containing external references will be marked on listings with a pointer next to the address and will not be treated as errors. Undefined labels will produce "U" error flags if SNAP/3 is producing absolute output. Undefined or forward referenced labels in the expression fields of directives other than TESTnn in pass one, DA, and DC will always produce "U" error flags.

The expression field is omitted for instructions which require no expression. The DA and DC directives accept multiple expressions delimited by commas (see sections 3.2 and 3.3).

There are twelve operators allowed in expressions:

2.3.1 + This means addition.

2.3.2 - This means subtraction. Note that the minus sign may be placed at the beginning of an expression if the value of the first item is to be negated.

2.3.3 * When used as the first character in the expression, this operator will set the assembler's star flag, which affects the evaluation of the expression, depending upon where it occurs (see sections 3.2, 3.3, 4.5, and 4.6). It may be followed by a minus operator (e.g. *-DOG+1). When used as the first character after a left parenthesis, it is ignored.

2.3.4 * When used between two operands, signifies 16-bit signed integer multiplication.

2.3.5 / A slash indicates signed integer division. Any remainder produced by the division will be ignored.

2.3.6 > This means shift right. The value accumulated up to this point is logically shifted right the number of places indicated in the following operand (all bits shifted off the end are discarded and zeros are filled in on the left). Negative numbers will be treated as unsigned 16-bit values instead of two's complement 16-bit values.

2.3.7 < This is the same as > except shifting is to the left with zero fill on the right.

2.3.8 .AND. This means to perform a logical "AND" of the two unsigned 16-bit numbers.

2.3.9 .OR. .IOR. These mean to perform a logical inclusive "OR" of the two unsigned 16-bit numbers.

2.3.10 .XOR. This means to perform a logical exclusive "OR" of the two unsigned 16-bit numbers.

2.3.11 .MOD. This means signed divide giving only the remainder produced by division.

Note that only the first character of a logical operation is used to determine the operation type and that additional characters prior to the second period are ignored.

2.4 Examples of expressions

The following examples assume that the value of DOG is 1 and that the value of CAT is 2.

VALID EXPRESSIONS	VALUE	
DOG	1	
DOG+1	2	
1+ DOG	2	
DOG+CAT	3	
'A'+1	0102	
*-CAT+1	-1	Note that star flag will be set.
-DOG<3	-8	
-DOG>3	8191	Note that sign is not extended on right shifts.
8>3+1	2	Note that shift occurs before addition.
CAT*CAT	4	
CAT.AND.DOG	0	
DOG.OR.CAT	3	
0377.XOR.DOG	0376	

ILLEGAL EXPRESSIONS

DOG+	Terminating character not a space or comma.
DOG#1	Illegal binary operator.
1 +DOG	Will not be flagged but +DOG will not be evaluated as part of the expression.
'AB'	Illegal if not a DC atatement. Only 1 character allowed in all other expression strings.
CAT+DOG=	Illegal terminator character.
CAT.NOT.1	Illegal binary operator.
**12	Star flag set but no multiplier exists for second asterisk.
.XOR.1	No value prior to operator.

2.5 Comment field

The comment field begins anywhere after the expression field, column 25 (if the expression field is not used), or column 2 (if column 1 contains a period, plus, or asterisk as noted in section 2.1). When placed following an instruction that does not use an expression, the comment field must not start prior to column 26. The comment field may contain any character and is terminated by the end of the line. SNAP/3 puts out its listing of the source line exactly as it is provided in the source code so formatting of comments will be maintained.

CHAPTER 3. SNAP/3 DIRECTIVES

Assembler Directives are used for setting the LOCATION COUNTER, ADDRESS COUNTER, and LABEL values to other than the normal sequential assignments and for defining constants. Other Directives are used to control certain SNAP/3 functions such as input file linking, source file assembly, program listing and macro definition. Note that forward and external references in the expression field are only permitted in TESTnn in pass one, DA, and DC directives.

3.1 Align Address

```
ALIGN   <exp>
```

Increments the LOCATION COUNTER and ADDRESS COUNTER until the LOCATION COUNTER is an even multiple of the expression value. The expression value must be a power of two (i.e. 2,4,8,16 etc.) or an "E" error will result. If the statement has a label, it will be given the value of the location counter after the ALIGN is performed. Will produce an "E" error if the LOCATION COUNTER PAB is not either absolute or required to start at the beginning of a page. If a LOC directive has specified "n" bytes per word, the ADDRESS COUNTER will be incremented by "n" times the amount the LOCATION COUNTER is incremented.

3.2 Define Address

```
DA      <exp>[,<exp>...]
```

Generates a two byte constant which is the address, LSB first, of each expression. Placing an * in front of an expression will cause the two bytes to be generated in the reverse order (MSB first, LSB second). For example:

```
DOG     EQU     01234
        DA      DOG,*DOG,1
```

gives the following octal values:

```
234 002 002 234 001 000
```

3.3 Define Constant

```
DC      <exp>[,<exp>...]
```

Generates eight bit object bytes from one or more expressions or strings found in the expression field delimited by commas. A leading asterisk (*) on any expression will produce two object bytes (LSB, MSB) and therefore addresses may be imbedded within DC directives. A special exception is made for string items found in the DC directive. All the characters of a string item are significant and as many words as necessary are generated to accommodate all the characters of the given string. This special string item is in effect only if the expression consists of only a string. String items in expressions still have only one character of significance. For example:

```
DC      1,2+3,'A'+2,'ABC'
```

generates the following octal values:

```
001,005,0103,0101,0102,0103
```

3.4 End

```
END     [<exp>]
```

Indicates that there is no more source code in the program to be processed and that SNAP/3 should proceed to the next pass, if any. The expression field has special significance in the END statement in that its value is taken as the Primary Transfer Address at which program execution will begin. This is optional and if the expression field is empty or no END statement is encountered, a Secondary Transfer Address is set by SNAP/3 to the location of the first byte of object code.

3.5 Equivalence

```
<label> EQU     <exp>
```

Sets the value of the label on the statement to the value of the expression field. Object code is not generated by EQU's, but dictionary labels are. One way of handling external references is by equating labels to the value of the external references and then referencing the labels. (A better way is usually to use LINK to resolve the external references.) Will produce an "E" error if

no label is found.

3.6 Error

ERR

Produce a "P" error flag. Usually follows a conditional assembly statement to trap a page or table overflow etc. For example:

```
TABLE   SK      LEN
        IFNE    $>8,TABLE>8
        ERR     TABLE OVERFLOWS A PAGE!
        XIF
```

3.7 IF

IFnn <exp>[,<exp>]

This is the conditional assembly directive. Condition "nn" (assumed to be "EQ" if not given) must be met in the signed comparison of the two expressions found separated by a comma in the expression field in order to assemble following lines of code. The second expression will be assumed zero if not given. Only an XIF directive will turn the conditional assembly back on. Any number of IF directives may occur before an XIF directive, but as soon as processing is turned off by one of the IF directives, the remaining IF directives will be ignored and processing will be turned back on by the first following XIF directive. An undefined or forward referenced expression operand is fatal and this occurrence will cause pass two to be aborted. The available condition codes are:

```
EQ    Field 1 must be equal to field 2
GT    Field 1 must be greater than field 2
LT    Field 1 must be less than field 2
NE    Field 1 must not be equal to field 2
NG    Field 1 must not be greater than field 2
NL    Field 1 must not be less than field 2
GE    Field 1 must be either greater than
      or equal to field 2
LE    Field 1 must be either less than
      or equal to field 2
Z     Field 1 must be zero
NZ    Field 1 must be non-zero
C     Field 1 must be clear
      (flag-testing, same as Z)
S     Field 1 must be set
      (flag-testing, same as NZ)
STR   Field 1 must begin with an asterisk (*)
NSTR  Field 1 must not begin with an asterisk
```

3.8 Include

```
INC     <filename>
```

Includes the source from filename specified in the expression field. The file specified may be in DOS format (as a free standing file,) or in library member format (filename/ext.member). Up to 62 files may be included. Lines of source code originating from an included file are noted by a trailing alphabetic character in the line number. Unused labels in included files are omitted from the "Unused Label" listing.

3.9 List

```
LIST    [-]<letter>[,...]
```

This is a directive which is used to alter the settings of SMAP/3's listing control flags. Each flag is specified by one character which turns the flag on when mentioned in a LIST statement, unless it is preceded by a minus sign (-) which will turn the flag off. Commas may be used to delimit more than one flag character. To allow nesting of listing control, a counter is associated with each flag. Whenever a LIST -x appears, the associated counter is incremented. Whenever a LIST x appears and the control flag is off, the counter is decremented, and the control flag is only turned on when LIST x has appeared as many times as LIST -x. The flag characters, their default settings,

and their usage are as follows:

L	ON	Master list control. If turned off, no pass two output will be listed until this flag is turned on again regardless of other control flags.
F	OFF	If-skipped lines. This flag must be on to produce a listing of all lines of source skipped by an IF<nn> statement.
G	OFF	Generated lines. If turned off, this flag will suppress the listing of code lines generated by DA, DC, and RPT statements.
I	OFF	Included lines. Lines of source code included from additional source files will not be listed unless this flag is on.
M	OFF	Macro expansion. This flag must be on to produce a listing of macro expansion source lines.

For example, LIST M,-I would turn on listing of expanded macros, but turn off listing of includes.

3.10 Location

```
LOC     <exp>[,<exp>]
LOC     *[,<exp>]
```

Sets LOCATION COUNTER to the value of expression field and sets the Located Mode flag. If the expression field consists of an asterisk (*), the Location flag is cleared and the LOCATION COUNTER is set to the ADDRESS COUNTER. If the statement has a label, it will be given the value of the location counter after the LOC is performed. Note that the listing will have the LOCATION COUNTER (noted by a trailing L) printed instead of the ADDRESS COUNTER while the Location flag is set. Remember that the LOCATION COUNTER indicates the address at which the code is to execute. If the expression is relocated by a relocatable PAB, then references to the current LOCATION COUNTER will be relocated by that PAB. The optional second expression is the number of bytes per word. This parameter is used when generating code for other machines whose word (address unit) size is larger than eight bits. If the value of this parameter is "n", the ADDRESS COUNTER will be incremented by "n" whenever the LOCATION COUNTER is incremented by one. A USE or SET directive resets the Location flag and resets the number of bytes per word to one.

3.11 Macro Definition

```
MACRO   [<exp>]
```

Indicates that the statements that follow are an inline definition of a macro prototype. (See Chapter 5.)

3.12 Macro Definition End

```
MEND
```

Marks the end of a macro definition. (See Chapter 5.)

3.13 Macro Library Include

```
MLIB    <filename>
```

Allows access to macros in the file specified in DOS format in the expression field. (See Chapter 5.)

3.14 Originate

```
<PAB>   ORG     <exp>[,<flag>[,...]]
```

Initializes a new Program Address Block (PAB) and sets its first and current word addresses to the value of the expression field. A PAB is relocatable if the expression given is zero. Following the address in the expression field, page alignment for relocatable PAB's is specified by ",T", ",P" and ",C". The "T" option generates a flag in the object code that tells the linkage editor to align the PAB at the beginning of a memory page. The "P" option generates an object code flag that tells the linkage editor to align the PAB so that it does not cross any memory page boundaries. The "C" option specifies that this PAB and all other PAB's with the same name are common and should be linked into the same area rather than being appended together. The label field defines the PAB's name which is referenced in the USE directive (section 3.24). It does not generate a label for the dictionary. A "D" error flag will be issued if the PAB has been previously defined; this is a fatal error.

3.15 Program Definition

<name> PROG

This is used to define the name to be used in the object code library to identify the program that follows. The label field gives the name of the segment produced. A PROG directive must be used in all but the first program when the source file being assembled by SNAP/3 contains more than one program. All object segments produced are placed in the same library.

3.16 Repeat

RPT <exp>

Will cause the following line of source code to be processed the number of times indicated by the LSB of the expression field's value. The following line may not be a RPT directive. For example:

```
RPT     5
CALL    INCHL
```

will produce the same code as:

```
CALL    INCHL
CALL    INCHL
CALL    INCHL
CALL    INCHL
CALL    INCHL
```

Repeating statements with labels which do not have a trailing = to signify a multiple definition will result in "D" error flags.

3.17 SET

SET <exp>

Clears the Location flag, initiates usage (USE) of the ABSOLUTE PAB (see section 3.24), and sets the ADDRESS COUNTER and LOCATION COUNTER to the value of the expression. If the statement has a label, it will be given the value of the location counter after the SET is performed.

3.18 SKIP

SKIP <exp>

Increments the values of the LOCATION COUNTER and ADDRESS COUNTER by the value of the expression field. The value may be positive or negative. If a LOC directive (section 3.10) has specified "n" bytes per word, then the ADDRESS COUNTER is incremented by "n" times the SKIP expression value.

3.19 Assembly Options

SNAPOPT <letter>[,<letter>...]

This is used to turn certain assembly options on or off during an assembly. Each option is specified by one character which turns the option on when mentioned in a SNAPOPT statement, unless it is preceded by a minus sign (-) which turns the option off. Each option is initially off at the beginning of each program unless the character was specified as an option on the command line, in which case the option is initially on. The options which may be specified on the SNAPOPT directive follow. See section 6.1 for a complete list of options.

U Instructions and pseudo-instructions for the 2200 and 5500 processors will not be defined. This permits these names to be defined as macros.

2 Only 2200 processor instructions are allowed. Instructions for the 5500 processor will produce an "I" error flag but will generate the correct code. This is usefull when assembling code to be executed on a 2200 processor.

6 Instructions for the 6600 processor are defined.

X This option only has effect if a cross reference listing was requested by the X option on the command line. Label definitions and references occurring while this option is off will not appear in the cross reference listing.

R This option only has effect if a cross reference listing was requested by the X option on the command line. If a label is defined while this option is on, then no references to that label occurring after the definition will appear in the cross reference listing. This is usefull if it is desired that certain labels not appear in the cross reference listing. This option may not appear on the SNAP3 command line.

H All numbers on the listing which are normally edited in octal will be hexadecimal instead. This option may not be dynamicly turned on and off throughout the listing; the state of the option at the end of the source code will be used throughout the listing.

B Generated object code bytes will be edited in binary instead of octal on the listing. This option may not be dynamicly turned on and off throughout the listing; the state of the option at the end of the source code will be used throughout the listing.

3.20 Test

TESTnn <exp>[,<exp>]

This directive tests whether the specified relation "nn" holds between the two operands. It differs from most other directives in that the operand expressions may contain forward references, and also in that the assembly must be a two pass assembly if this directive is used. The assembly will be two pass if a source listing was requested by the D or L option on the command line or if absolute output was requested by the A option, or two passes may be forced by the T option. This directive will produce an "E" flag if the specified condition is not met, if the assembly is not two pass, or if the value of either expression is relocatable. The possible relations are the same as for the IF directive (section 3.7) except for omitted, STR, and NSTR. For example:

```
TESTGE  ABC-$,-128

TESTLE  ABC-$,127
```

would produce an error flag if the label ABC were not within the range [$-128,$+127].

3.21 Title

TITLE

Causes the program listing to page eject and print the page heading followed by text taken from the line immediately following the TITLE statement. The title will continue to print at the top of each page until changed by another TITLE directive.

3.22 Tabulate Maybe

TM <exp>

Performs a Tabulate Page (section 3.23) if the value of the expression field would cause a page overflow if added to the current LOCATION COUNTER. If the statement has a label, it will be given the value of the location counter after the TM is performed. Will produce an "E" error if the LOCATION COUNTER PAB is not either absolute or required to start at the beginning of a page.

3.23 Tabulate Page

TP

Increments the value of ADDRESS COUNTER and the the LOCATION COUNTER until the LOCATION COUNTER value is a multiple of 256 (LSB = 000). This is useful for setting up page-dependent data areas which are addressable by single precision (leaving H fixed and manipulating only the L-register). If the statement has a label, it will be given the value of the location counter after the TP is performed. Will produce an "E" error if the LOCATION COUNTER PAB is not either absolute or required to start at the beginning of a page. If a LOC directive (section 3.10) has specified "n" bytes per word, then the ADDRESS COUNTER is incremented by "n" times the amount the LOCATION COUNTER is incremented.

3.24 Usage

USE <PAB>

Initiates usage of the PAB whose name is given in the expression field. An asterisk (*) in the expression field will revert back to the last PAB used. If the statement has a label, it will be given the value of the location counter after the USE

is performed. A "U" error will be issued if the PAB named has not been defined by an ORG statement; this is a fatal error.

3.25 XIF

```
XIF
```

Force the assembly on if it has been conditionally turned off.

CHAPTER 4. PSEUDO-INSTRUCTIONS

Pseudo-instructions are predefined mnemonics for commonly used instruction sequences. They cause SNAP/3 to generate a sequence of machine instructions to perform the desired function.

4.1 HL

HL <exp>

The HL pseudo-instruction generates the load H register and load L register instructions necessary to place the value of the expression field in the H and the L registers properly, so that a load to or from memory will use that address, i.e., H contains the MSB and L contains the LSB. The HL pseudo-instruction generates four bytes of object code. For example:

```
OOPS  EQU  02005
      HL   OOPS
```

generates the following code:

```
066 005 056 004
```

4.2 DE

DE <exp>

The DE pseudo-instruction works the same as the HL pseudo-instruction except it loads the D and E registers instead of H and L.

4.3 BC

BC <exp>

The BC pseudo-instruction works the same as the HL pseudo-instruction except it loads the B and C registers instead of H and L.

4.4 XA

XA <exp>

The XA pseudo-instruction works the same as the HL pseudo-instruction except it loads the X and A registers instead of H and L.

4.5 Memory Store

MSr [*]<exp>

The Memory Store pseudo-instruction allows the user to store a given register into a given memory location. Placing an * in front of the expression causes the H-register to be loaded as well as the L. The expansion is as follows:

```
LL      <exp>
LH      <exp>>>8 if * is present
LMr
```

4.6 Memory Load

MLr [*]<exp>

The Memory Load pseudo-instruction works the same as Memory Store (MSr) with the exception that the register is loaded from memory rather than being stored into memory.

4.7 Shift Right

SRN <exp>

The Shift Right numeric pseudo-instruction allows the user to generate SRC instructions the number of times specified in the expression field. The expression must be defined in pass one and must have a value between zero and seven. For example:

```
SRN  3
```

will generate the following code:

```
012 012 012
```

4.8 Shift Left

SLN <exp>

The Shift Left numeric pseudo-instruction works the same as SRN with the exception that SLC instructions (002) are generated.

4.9 Condition Code Load

CCL[r]

The Condition Code Load pseudo-instruction generates an ADrr instruction (ADA if r is omitted) which will reload the condition code after it has been saved in register r by a CCS (condition code save) instruction or equivalent.

CHAPTER 5. MACROS

Macros are predefined sections of source code which may be used to facilitate the coding of commonly used procedures. Macro source code is modified by SNAP/3 to include labels and expressions passed as arguments by the main body of source statements.

Macro definitions are called "Macro Prototypes" and are saved for later access by the SNAP/3 assembler.

5.1 Preparing Macro Prototypes

The DOS editor is used to produce prototype statements. Macro prototypes must be entered in the following format:

```
        MACRO  [expression]
[label] name   [symbol[(default)]][,symbol[(default)]]..etc.
[one or more assembly-language statements]
        MEND
```

Each prototype must start with a statement with "MACRO" in the instruction field and end with a statement with "MEND" in the instruction field. The optional expression on the MACRO line specifies the number of parameter lists on the second line as described below.

The second statement of each prototype is called a "Macro Prototype Header" and defines the name of the macro and any labels and symbols that may be replaced during assembly. The name may be any 1 to 8 character symbol that is not already predefined by SNAP/3 as an instruction mnemonic or assembly directive (See Appendix C). All arguments shown in brackets are optional and may be omitted if not needed.

Labels and symbols shown in the prototype header define items in the statements that follow that may be replaced at assembly time. Following each symbol in the header a default expression may be defined. The default will be used if a macro reference in SNAP/3 fails to supply a replacement expression for the preceding symbol.

The operand field of the prototype header consists of one or more lists separated by blanks, with each list consisting of one or more symbols (with defaults) separated by commas (,). If more than one list is present, the number of lists must be specified as the operand field of the MACRO line. A "zeroth" list may also be supplied, separated from the prototype header name by a comma; this list is not counted in the number specified on the MACRO line.

One or more macro definitions may be defined in the same file using the DOS editor. Macro definitions may occur in line in the same program in which they are to be used, or they may be placed in macro libraries, which are created by the LIB utility.

5.2 Macro Calls

Code from a macro prototype library is included in SNAP/3 assemblies by the means of "macro calls". Each library containing macros to be included must first be made known to the assembler by means of a MLIB directive. The MLIB directive is entered in the instruction field followed by the macro library file name in the following format:

```
MLIB    file-name
```

If the file-name's extension is omitted, /MPL will be assumed.

Macro calls are coded as follows:

```
[label]  name    [expression][,expression]...etc.
```

The name used in the instruction field will be assumed to be a macro name if is not a recognizable SNAP/3 instruction mnemonic or assembly directive (See Appendix C). The label and expression arguments in brackets are optional. Arguments defined in the expression field are positional and must be defined in the same order as related symbols in the macro's prototype header.

5.3 Macro Definitions within Programs

Macros may be defined in the same program in which they are to be used by simply defining macro prototypes prior to their first reference by a macro call in the program. The macros may be defined in the source file or an INCluded file.

When macros are defined inline, a MLIB statement is not required for their use within the assembly.

5.4 Macro Expansion

Note the similarity between the format of a macro call and macro prototype header. They are closely related and determine the final code that will be included in your assembly.

```
Call:   [label] name [expression][,expression] [expression]..etc.
Header: [label] name [sym[(def)]][,sym[(def)]] [sym[(def)]]..etc.
```

The label for the call will replace the occurrences of the header label in prototype code during expansion. The first expression in the call will replace the first header symbol in the prototype code, the second expression will replace the second symbol, and so forth.

Arguments may be omitted in each list of macro call expressions by coding only the trailing comma to indicate the missing expression. Trailing commas after the last expression in a list are not required.

The rules for substitution are:

Macro Call	Prototype Header	Action
Label	No Label	Label is defined normally before expanded macro code is processed.
Label	Label	Call label substituted in expanded macro code.
No label	No label	No change.
No label	Label	Prototype label is unchanged.
Symbol	No symbol	Call symbol ignored.
Symbol	Symbol	Call expression substituted for occurrences in macro code.
No symbol	Symbol but no default	Header symbol disappears in expanded code.
No symbol	Symbol with a default	Default substituted for occurrences in macro code.

Symbols within apostrophes (') are never replaced during expansion. Substitution of arguments is best shown by example:

Macro call:

```
LOOP     CLEAR  BUFFER
```

Macro prototype:

```
         MACRO
LABEL    CLEAR  FIELD,SIZE(80)
         HL     FIELD
         LB     0
         LC     SIZE
LABEL    LMB            CLEAR THE FIELD
         INCP   HL
         SUC    1       DECREMENT COUNT
         JFZ    LABEL   CONTINUE
         MEND
```

Expansion:

```
         HL     BUFFER
         LB     0
         LC     80
LOOP     LMB            CLEAR THE BUFFER
         INCP   HL
         SUC    1       DECREMENT COUNT
         JFZ    LOOP    CONTINUE
```

You will note in the preceding example that the symbols "LABEL" and "FIELD" in the prototype have been replaced by "LOOP" and "BUFFER" provided by the macro call. The symbol "SIZE" did not have a replacement expression in the macro call and the default "80" has been substituted.

5.5 Global Labels

Global labels are labels which can be referenced anywhere in a SNAP/3 assembly. Each global label name must be unique within an assembly since references may occur in both the main code as well as within macro expansion code.

Any label in the label field of any line of a macro prototype that is altered or replaced by a macro call argument or macro

prototype default automatically becomes global.

In the preceding example, the label "LOOP" is global.

5.6 Local Labels

Local labels are labels which can be referenced only within the macro expansion in which they occur. Each macro expansion generates an identifying number which is associated internally with all local labels within the current expansion. Local label names may be duplicated many times within an assembly, however, SNAP/3 considers each unique to the macro expansion in which it occurred.

Any label in the label field of any line of a macro prototype that is not altered or replaced during macro expansion is automatically declared a local label.

For example:

Macro-prototype:

```
         MACRO  2
         COUNT  AAA,BBB(0)  CCC
         BC     BBB
OUTER    DE     CCC
INNER    DECP   DE
         JFC    INNER
AAA      DECP   BC
         JFC    OUTER
         MEND
```

Macro Calls followed by Expansions:

```
NEXT     COUNT  ,1000 500
```

```
NEXT     BC     1000
OUTER    DE     500
INNER    DECP   DE
         JFC    INNER
         DECP   BC
         JFC    OUTER
```

```
         COUNT  BCSET 999
```

```
         BC     0
OUTER    DE     999
INNER    DECP   DE
         JFC    INNER
BCSET    DECP   BC
         JFC    OUTER
```

In the example on the left, a global label line is generated for NEXT. OUTER and INNER become local labels and the symbol AAA in the prototype disappears.

In the example on the right, OUTER and INNER become local labels and the symbol AAA in the prototype becomes the global label BCSET.

5.7 Macro Nesting

SNAP/3 allows nesting of macros calls within macro calls with up to eight levels of expansion. Local labels cannot be passed as arguments to inner macros, however, passage of global labels and other arguments is unrestricted.

For example:

```
         MACRO                          INLINE DEFINITION 1
         LEVEL1 ARG
XXX      DC     ARG
         LEVEL2 ARG
         DA     XXX
         MEND
.
         MACRO                          INLINE DEFINITION 2
         LEVEL2 ARG
XXX      DC     ARG+1
         DA     XXX
         MEND
.
XXX      DC     0                       MAINLINE CODE
         LEVEL1 017
         DA     XXX
```

Expands to:

```
14. 000000  000          XXX    DC      0        MAINLINE CODE
15.                             LEVEL1  017
15. 000001  017          XXX    DC      017
15.                             LEVEL2  017
15. 000002  020          XXX    DC      017+1
15. 000003  002 000             DA      XXX
15. 000005  001 000             DA      XXX
16. 000007  000 000             DA      XXX
```

In the preceding example, XXX is defined as a label three times. The first definition is a global label in the main body of code. The second and third definitions are as local labels at different levels of macro expansion.

5.8 Forcing characters

The "at" sign (@) is used in macro call and macro prototype expressions as a forcing character. Its primary purpose is to allow blanks, commas, apostrophes, and concatenation characters to be transferred to an expansion without evaluation.

Forcing characters are not transferred to the expanded code.

For example:

Macro-prototype:

```
        MACRO
        STRING A,B
        DC     A,@'B@'
        MEND
```

Macro Call and Expansion:

```
        STRING 100@,200@,300,ERROR@ MESSAGE@ ##1
        DC     100,200,300,'ERROR MESSAGE ##1'
```

Note that in the macro prototype, forcing characters are used to prevent evaluation of apostrophes and allow a substitution to be made between them (substitutions are normally suppressed between apostrophes). Commas and blanks have been forced in the expression field of the macro call to prevent their being interpreted as expression delimiters.

In the macro call, "100@,200@,300" is considered to be one expression and "ERROR@ MESSAGE@ ##1" is considered to be a second expression.

5.9 Concatenation

The concatenation character (¦) is used in inner macro calls and macro prototype expressions to separate symbols into individually replaceable elements. During macro expansion, concatenation characters in the expression field that are not within apostrophes or preceded by a forcing character will be omitted from the generated code.

For example:

Macro-prototype:

```
         MACRO
         MSG     AAA,BBB
MSG¦AAA  DA      BBB¦LOC,BBB¦SIZE
         DC      @'ERROR@ IN@ PHASE@ DCT¦AAA@'
         MEND
```

Macro Call and Expansion:

```
         MSG     024,PHS4
MSG024   DA      PHS4LOC,PHS4SIZE
         DC      'ERROR IN PHASE DCT024'
```

5.10 Macro Directives

Macro directives provide a means of conditionally generating lines of macro code depending on what replacement expressions have been specified for prototype symbols. Macro directives are evaluated and executed during macro expansion.

5.10.1 Macro IF

Macro IF directives are coded in the following format:

```
MIFnn  string1[,string2]
```

Macro code following a MIF is generated only when the selected condition (nn) is found to be true:

Directive	True Condition
MIF	String1 is set (Not null)
MIFS	String1 is set (Not null)
MIFC	String1 is clear (Null)
MIFLT	String1 is less than string2
MIFEQ	String1 is equal to string2
MIFGT	String1 is greater than string2

MIFLE	String1 is less than or equal to string2
MIFGE	String1 is greater than or equal to string2
MIFNL	String1 is not less than string2
MIFNE	String1 is not equal to string2
MIFNG	String1 is not greater than string2

A string2 should not be specified for MIF, MIFS and MIFC. If specified it will be ignored.

A string2 must be specified for MIFLT, MIFEQ, MIFGT, MIFLE, MIFGE, MIFNL, MIFNE, and MIFNG. If string2 is not specified, it is assumed to be a null string. Strings of characters specified for comparison are terminated by the first blank or comma character that is not preceded by a forcing character or within apostrophes. If the two strings are different length but otherwise equal, the shorter string is considered to be less.

For Example:

```
        MIFEQ  PAR1¦PAR2,ABC' DEFG '@ COMMENT HIJKL
```

The second comparison string starts with the letter A and ends with the letter T.

5.10.2 Macro IF Exit

Conditional generation of macro code is terminated by a Macro-eXit-IF (MXIF) directive.

For example:

```
        MACRO
        TEST   P1
        MIFEQ  P1,ASCII
. THIS COMMENT WILL GENERATE IF P1 IS ASCII
        MXIF
        MIFNE  P1,ASCII
. THIS COMMENT WILL GENERATE IF P1 IS NOT ASCII
        MXIF
. THIS WILL GENERATE UNCONDITIONALLY
        MEND
```

CHAPTER 6. OPERATING PROCEDURES

The DOS command requesting execution of the SNAP/3 assembler should be as follows:

SNAP3 source[,object][,ept][,print][,include][;<option chars.>]

where each bracketed object and each character after the semicolon is optional.

6.1 Parameterization

The first file specification (which is required) is the source file, the second file specification is for the object file, the third file specification is for the entry point file, the fourth specification is the print file, and the fifth specification is a file which will be INCluded (see section 3.8) before the source file is processed. The source file has a default extension of TXT. The object file, if not given, is assumed to have the same name as the source file and has a default extension of REL if relocatable output is being produced, ABS if absolute output is being produced. The entry point file name, if not given, is assumed to have the same name as the program name (which defaults to the object file name if there is no PROG directive). The entry point file has a default extension of EPT and a default drive the same as the drive the object file is written on, unless the entry point file already exists. The entry point file is written after pass one only if entry points have been declared in the program. The EPT file is written in a compressed symbolic format which can be INCLUDED by a later assembly to provide a program linking capability. The print file has a default name the same as the object file name and a default extension of PRT. The include file name has a default extension of TXT.

The characters on the command line following the semicolon select SNAP/3 options. The following options may be specified:

A — Causes an absolute output file to be produced, instead of a relocatable file.

D — Causes a source and object code listing to be displayed on the CRT; may be specified in addition to the L option.

F,G,I,M Turns on corresponding listing control flags (see section 3.9).

L Produces a source and object code listing. The listing will be on the local printer if neither the P, Q, nor S option appears.

P Causes the L or X option listing to be to a print file.

Q Same as P option, but specifies that the listing should be appended or queued after any information already in the print file.

S Causes the L or X option listing to be to the servo printer.

T Forces a two pass assembly. Must be specified if the relocatable output file produced is to be loaded by the DOS relocatable loader (DOS function 15).

X Produces a cross-reference map listing. May appear with or without the L option.

? This causes a list of options and the command line format to be displayed. No assembly is done.

2,6,B,H,U Turns on the assembly options described in section 3.19.

6.2 SNAP/3 Pass One

Initially SNAP/3 will validate the file specifications and the options selected. The version and revision numbers identifying the release will be displayed. If P or Q appeared on the command line but the print file was not specified, the print file specification is requested. The default file name is the object file name, and the default extension is PRT. The program will request an 80-character heading if either the L or X parameter has been specified. SNAP/3 will then read the source file and any INCLUDED files in order to build a dictionary containing all symbolic names used by the programmer and their equivalent octal value or address. A notation is printed as each INCLUDE is processed along with any lines which contain errors.

At the end of pass one, one or more of the following items

will be displayed on the CRT:

1) Any pass one error flags
2) Fatal error message if fatal error occurred
3) Program Address Blocks--name, origin, and length
4) Primary Transfer Address--octal value

If a program listing has been requested, one or more of the following items will be printed on the printer device or CRT:

1) Any pass one error flags
2) Fatal error message if fatal error occured
3) Program Address Blocks--name, origin, and length
4) Primary Transfer Address--octal value
5) Entry Points--name, value
6) External definitions--name, value
7) External references
8) Unused labels
9) Multiply defined labels

6.3 SNAP/3 Pass Two

If no fatal pass one errors occurred, SNAP/3 will now write the entry point file, if required, and proceed into pass two, if required. Pass two is responsible for the resolution of forward references and the generation of a program listing.

6.4 Cross-Reference Generation

At the completion of pass two, SNAP/3 will call the DOS SORT if a cross-reference listing is desired. DOS SORT will sort the label definitions and references and write a sorted label file. It will then overlay itself with SNAP/3 which will list the sorted references.

The actual listing of references will contain the symbolic name preceded by its actual octal value. Following the symbolic name is a list of all line numbers at which that symbolic name was defined or referenced. All definition lines are flagged with a leading asterisk while all Inclusions are noted by a trailing colon followed by the Inclusion file character (see section 3.8). Macro internal labels will have (M) after their name and each usage and associated references will be grouped and listed separately. Duplicate references with the same line number will be suppressed. For example:

```
11304 DECHL        *32:A   *32:B

00341 DISPL        *24

00024 IDLE        *197     212

00035 INDEX  (M)  *904     900     906

00057 INDEX  (M)  *913     936

10176 INCHL       *102      71     151     156

00007 MANY         *25:A   *25:B    21:A    21:B
```

If a symbol has duplicate definitions, the octal value shown is the initial value assigned.

6.5 Assembly Errors

SNAP/3 produces error flags to indicate source program errors. Some serious errors are fatal; these cause the second assembly pass to be skipped and any active CHAIN to be aborted. The other errors set the ABTIF flag which can be tested in a CHAIN (see the DOS manual). The fatal errors are mentioned in the sections describing constructions which can cause them.

The ERROR FLAGS produced by SNAP/3 are as follows:

6.5.1 D The D flag means DUPLICATE DEFINITION. It is generated if an attempt has been made to define the label more than once without a trailing = mark. Note that a reference to a duplicately defined label will use the most recent previous definition, or the last definition if the label has not been previously defined.

6.5.2 E The E flag means that an error has occurred in an EXPRESSION or some unrecognizable character appeared in the wrong place. In this case, a zero is substituted for the expression or for whatever was unrecognizable if code generation was expected.

6.5.3 F The F flag means FILE error. It can be issued for an INC or MLIB directive because the specified file is not found.

6.5.4 I The I flag means INSTRUCTION MNEMONIC UNDEFINED. The instruction was not an acceptable instruction and three octal zeroes are inserted for the instruction.

6.5.5 O The O flag means memory page OVERFLOW. It is issued when generated code in a page restricted Program Address Block crosses a memory page boundary.

6.5.6 P The P flag means PROGRAMMER PRODUCED. It is issued when an ERR directive is processed.

6.5.7 U The U flag means UNDEFINED LABEL. It is issued in pass two whenever a label is referenced and is not defined if absolute output is being produced. It is also issued when an assembly directive in pass one (except DA, or DC, or TESTnn) is operating on an expression containing a label not yet in the dictionary. Other undefined symbols in relocatable assemblies are assumed to be external references, and are marked with ">" on the listing.

6.6 DISPLAY and KEYBOARD Keys

The DISPLAY key may be depressed at any time to cause SNAP/3 to pause while displaying data. Normal processing will resume when the DISPLAY key is released.

The KEYBOARD key may be depressed at any time to cause SNAP/3 to abort the assembly.

6.7 Temporary Files

SNAP/3 may use up to four temporary files, plus any temporary files used by the SORT utility if a cross reference is requested. These files are placed on the same drive as the object file, and are deleted at the end of the assembly. The files are: SNPTEMPn/SYS if the assembly is two pass, SNPPAGEn/SYS if SNAP/3's working tables will not all fit in memory, and SNPXREFn/SYS and SNPSXRFn/SYS if a cross reference is requested. The character 'n' in the file names will be '0' if the Partition Supervisor (PS) is not active or the partition identifier if it is.

APPENDIX A. ASCII-OCTAL EQUIVALENTS

The standard octal equivalents for the ASCII character set. Interpretations will vary with some printers and display devices.

A 101	a 141	0 060	: 072
B 102	b 142	1 061	; 073
C 103	c 143	2 062	< 074
D 104	d 144	3 063	= 075
E 105	e 145	4 064	> 076
F 106	f 146	5 065	? 077
G 107	g 147	6 066	[133
H 110	h 150	7 067	~ 176
I 111	i 151	8 070	] 135
J 112	j 152	9 071	^ 136
K 113	k 153	Space 040	_ 137
L 114	l 154	! 041	@ 100
M 115	m 155	" 042	{ 173
N 116	n 156	# 043	\ 134
O 117	o 157	$ 044	¦ 174
P 120	p 160	% 045	} 175
Q 121	q 161	& 046	
R 122	r 162	' 047	
S 123	s 163	(050	
T 124	t 164	) 051	
U 125	u 165	* 052	
V 126	v 166	+ 053	
W 127	w 167	, 054	
X 130	x 170	- 055	
Y 131	y 171	. 056	
Z 132	z 172	/ 057	

APPENDIX B. DATAPOINT 2200/5500/6600 INSTRUCTION MNEMONICS

The following is a list of all Datapoint processor instruction mnemonics accepted by SNAP/3 with the octal code generated for each instruction.

In the instruction expression field the following abbreviations are used:

data	- immediate data
loc	- location
disp	- displacement

In the generated code the following abbreviations are used:

vvv	- 8 bits of immediate data
lsb	- least significant 8 bits of location or displacement
msb	- most significant 8 bits of location or displacement
ndx	- least significant 8 bits of index (msb in X)

In the description the following abbreviations are used:

data	- 8 bits of immediate data in instruction code (vvv)
A-E,H,L,X	- contents of the specified register
(BC),(DE),(HL),(XA)	- contents of memory pointed to by register pair

AC	data	014 vvv	Add with carry data to A
ACA		210	Add with carry A to A
ACA	data	014 vvv	Add with carry data to A
ACAA		210	Add with carry A to A
ACAB		111 210	Add with carry A to B
ACAC		062 210	Add with carry A to C
ACAD		113 210	Add with carry A to D

ACAE		174 210		Add with carry A to E
ACAH		115 210		Add with carry A to H
ACAL		176 210		Add with carry A to L
ACAX		117 210		Add with carry A to X
ACB		211		Add with carry B to A
ACB	data	111 014	vvv	Add with carry data to B
ACBA		211		Add with carry B to A
ACBB		111 211		Add with carry B to B
ACBC		062 211		Add with carry B to C
ACBD		113 211		Add with carry B to D
ACBE		174 211		Add with carry B to E
ACBH		115 211		Add with carry B to H
ACBL		176 211		Add with carry B to L
ACBX		117 211		Add with carry B to X
ACC		212		Add with carry C to A
ACC	data	062 014	vvv	Add with carry data to C
ACCA		212		Add with carry C to A
ACCB		111 212		Add with carry C to B
ACCC		062 212		Add with carry C to C
ACCD		113 212		Add with carry C to D
ACCE		174 212		Add with carry C to E
ACCH		115 212		Add with carry C to H
ACCL		176 212		Add with carry C to L
ACCX		117 212		Add with carry C to X
ACD		213		Add with carry D to A
ACD	data	113 014	vvv	Add with carry data to D
ACDA		213		Add with carry D to A
ACDB		111 213		Add with carry D to B
ACDC		062 213		Add with carry D to C
ACDD		113 213		Add with carry D to D
ACDE		174 213		Add with carry D to E
ACDH		115 213		Add with carry D to H
ACDL		176 213		Add with carry D to L
ACDX		117 213		Add with carry D to X
ACE		214		Add with carry E to A
ACE	data	174 014	vvv	Add with carry data to E
ACEA		214		Add with carry E to A
ACEB		111 214		Add with carry E to B
ACEC		062 214		Add with carry E to C
ACED		113 214		Add with carry E to D
ACEE		174 214		Add with carry E to E
ACEH		115 214		Add with carry E to H
ACEL		176 214		Add with carry E to L
ACEX		117 214		Add with carry E to X

ACH		215	Add with carry H to A
ACH	data	115 014 vvv	Add with carry data to H
ACHA		215	Add with carry H to A
ACHB		111 215	Add with carry H to B
ACHC		062 215	Add with carry H to C
ACHD		113 215	Add with carry H to D
ACHE		174 215	Add with carry H to E
ACHH		115 215	Add with carry H to H
ACHL		176 215	Add with carry H to L
ACHX		117 215	Add with carry H to X
ACL		216	Add with carry L to A
ACL	data	176 014 vvv	Add with carry data to L
ACLA		216	Add with carry L to A
ACLB		111 216	Add with carry L to B
ACLC		062 216	Add with carry L to C
ACLD		113 216	Add with carry L to D
ACLE		174 216	Add with carry L to E
ACLH		115 216	Add with carry L to H
ACLL		176 216	Add with carry L to L
ACLX		117 216	Add with carry L to X
ACM		217	Add with carry (HL) to A
ACMA		217	Add with carry (HL) to A
ACMB		111 217	Add with carry (HL) to B
ACMC		062 217	Add with carry (HL) to C
ACMD		113 217	Add with carry (HL) to D
ACME		174 217	Add with carry (HL) to E
ACMH		115 217	Add with carry (HL) to H
ACML		176 217	Add with carry (HL) to L
ACMX		117 217	Add with carry (HL) to X
ACX	data	117 014 vvv	Add with carry data to X
AD	data	004 vvv	Add data to A
ADA		200	Add A to A
ADA	data	004 vvv	Add data to A
ADAA		200	Add A to A
ADAB		111 200	Add A to B
ADAC		062 200	Add A to C
ADAD		113 200	Add A to D
ADAE		174 200	Add A to E
ADAH		115 200	Add A to H
ADAL		176 200	Add A to L
ADAX		117 200	Add A to X
ADB		201	Add B to A

ADB	data	111	004 vvv	Add data to B
ADBA		201		Add B to A
ADBB		111	201	Add B to B
ADBC		062	201	Add B to C
ADBD		113	201	Add B to D
ADBE		174	201	Add B to E
ADBH		115	201	Add B to H
ADBL		176	201	Add B to L
ADBX		117	201	Add B to X
ADC		202		Add C to A
ADC	data	062	004 vvv	Add data to C
ADCA		202		Add C to A
ADCB		111	202	Add C to B
ADCC		062	202	Add C to C
ADCD		113	202	Add C to D
ADCE		174	202	Add C to E
ADCH		115	202	Add C to H
ADCL		176	202	Add C to L
ADCX		117	202	Add C to X
ADD		203		Add D to A
ADD	data	113	004 vvv	Add data to D
ADDA		203		Add D to A
ADDB		111	203	Add D to B
ADDC		062	203	Add D to C
ADDD		113	203	Add D to D
ADDE		174	203	Add D to E
ADDH		115	203	Add D to H
ADDL		176	203	Add D to L
ADDX		117	203	Add D to X
ADE		204		Add E to A
ADE	data	174	004 vvv	Add data to E
ADEA		204		Add E to A
ADEB		111	204	Add E to B
ADEC		062	204	Add E to C
ADED		113	204	Add E to D
ADEE		174	204	Add E to E
ADEH		115	204	Add E to H
ADEL		176	204	Add E to L
ADEX		117	204	Add E to X
ADH		205		Add H to A
ADH	data	115	004 vvv	Add data to H
ADHA		205		Add H to A
ADHB		111	205	Add H to B
ADHC		062	205	Add H to C

ADHD	113 205		Add H to D
ADHE	174 205		Add H to E
ADHH	115 205		Add H to H
ADHL	176 205		Add H to L
ADHX	117 205		Add H to X
ADL	206		Add L to A
ADL data	176 004	vvv	Add data to L
ADLA	206		Add L to A
ADLB	111 206		Add L to B
ADLC	062 206		Add L to C
ADLD	113 206		Add L to D
ADLE	174 206		Add L to E
ADLH	115 206		Add L to H
ADLL	176 206		Add L to L
ADLX	117 206		Add L to X
ADM	207		Add (HL) to A
ADMA	207		Add (HL) to A
ADMB	111 207		Add (HL) to B
ADMC	062 207		Add (HL) to C
ADMD	113 207		Add (HL) to D
ADME	174 207		Add (HL) to E
ADMH	115 207		Add (HL) to H
ADML	176 207		Add (HL) to L
ADMX	117 207		Add (HL) to X
ADX data	117 004	vvv	Add data to X
ALPHA	030		Select Alpha mode
BCP	041		Block compare
BCV	062 021		Block convert
BETA	020		Select Beta mode
BFAC	011		Binary field add with carry
BFLRAD	111 006		Binary field left to right add
BFLRAC	111 016		Binary field left to right add with carry
BFLRSU	111 026		Binary field left to right subtract
BFLRSB	111 036		Binary field left to right subtract with borrow

BFLRND	111 046	Binary field left to right and
BFLRXR	111 056	Binary field left to right exclusive or
BFLROR	111 066	Binary field left to right or
BFSB	031	Binary field subtract with borrow
BFSL	075	Binary field shift left
BFSR	111 075	Binary field shift right
BP	052	Break point
BRL	072	Base register load from A
BRLA	072	Base register load from A
BRLB	111 072	Base register load from B
BRLC	062 072	Base register load from C
BRLD	113 072	Base register load from D
BRLE	174 072	Base register load from E
BRLH	115 072	Base register load from H
BRLL	176 072	Base register load from L
BRLX	117 072	Base register load from X
BT	021	Block transfer
BTR	111 021	Block transfer reverse
CALL loc	106 lsb msb	Subroutine call
CCS	042	Condition code save in A
CCSA	042	Condition code save in A
CCSB	111 042	Condition code save in B
CCSC	062 042	Condition code save in C
CCSD	113 042	Condition code save in D
CCSE	174 042	Condition code save in E
CCSH	115 042	Condition code save in H
CCSL	176 042	Condition code save in L
CCSX	117 042	Condition code save in X
CFC loc	102 lsb msb	Subroutine call if false carry
CFB loc	102 lsb msb	Subroutine call if false borrow
CFZ loc	112 lsb msb	Subroutine call if false zero
CFE loc	112 lsb msb	Subroutine call if false equal
CFS loc	122 lsb msb	Subroutine call if false

				sign
CFL	loc	122	lsb msb	Subroutine call if false less
CFN	loc	122	lsb msb	Subroutine call if false negative
CFP	loc	132	lsb msb	Subroutine call if false parity
COMP	BC	062	011	2's complement BC
COMP	DE	174	011	2's complement DE
COMP	HL	176	011	2's complement HL
COMPS	BC	113	011	2's complement BC
COMPS	DE	115	011	2's complement DE
COMPS	HL	117	011	2's complement HL
CP	data	074	vvv	Compare A to data
CPA		270		Compare A to A
CPA	data	074	vvv	Compare A to data
CPAA		270		Compare A to A
CPAB		111	270	Compare B to A
CPAC		062	270	Compare C to A
CPAD		113	270	Compare D to A
CPAE		174	270	Compare E to A
CPAH		115	270	Compare H to A
CPAL		176	270	Compare L to A
CPAX		117	270	Compare X to A
CPB		271		Compare A to B
CPB	data	111	074 vvv	Compare B to data
CPBA		271		Compare A to B
CPBB		111	271	Compare B to B
CPBC		062	271	Compare C to B
CPBD		113	271	Compare D to B
CPBE		174	271	Compare E to B
CPBH		115	271	Compare H to B
CPBL		176	271	Compare L to B
CPBX		117	271	Compare X to B
CPC		272		Compare A to C
CPC	data	062	074 vvv	Compare C to data
CPCA		272		Compare A to C
CPCB		111	272	Compare B to C
CPCC		062	272	Compare C to C
CPCD		113	272	Compare D to C
CPCE		174	272	Compare E to C
CPCH		115	272	Compare H to C
CPCL		176	272	Compare L to C

CPCX		117 272		Compare X to C
CPD		273		Compare A to D
CPD	data	113 074	vvv	Compare D to data
CPDA		273		Compare A to D
CPDB		111 273		Compare B to D
CPDC		062 273		Compare C to D
CPDD		113 273		Compare D to D
CPDE		174 273		Compare E to D
CPDH		115 273		Compare H to D
CPDL		176 273		Compare L to D
CPDX		117 273		Compare X to D
CPE		274		Compare A to E
CPE	data	174 074	vvv	Compare E to data
CPEA		274		Compare A to E
CPEB		111 274		Compare B to E
CPEC		062 274		Compare C to E
CPED		113 274		Compare D to E
CPEE		174 274		Compare E to E
CPEH		115 274		Compare H to E
CPEL		176 274		Compare L to E
CPEX		117 274		Compare X to E
CPH		275		Compare A to H
CPH	data	115 074	vvv	Compare H to data
CPHA		275		Compare A to H
CPHB		111 275		Compare B to H
CPHC		062 275		Compare C to H
CPHD		113 275		Compare D to H
CPHE		174 275		Compare E to H
CPHH		115 275		Compare H to H
CPHL		176 275		Compare L to H
CPHX		117 275		Compare X to H
CPL		276		Compare A to L
CPL	data	176 074	vvv	Compare L to data
CPLA		276		Compare A to L
CPLB		111 276		Compare B to L
CPLC		062 276		Compare C to L
CPLD		113 276		Compare D to L
CPLE		174 276		Compare E to L
CPLH		115 276		Compare H to L
CPLL		176 276		Compare L to L
CPLX		117 276		Compare X to L
CPM		277		Compare A to (HL)
CPMA		277		Compare A to (HL)

CPMB		111 277	Compare B to (HL)
CPMC		062 277	Compare C to (HL)
CPMD		113 277	Compare D to (HL)
CPME		174 277	Compare E to (HL)
CPMH		115 277	Compare H to (HL)
CPML		176 277	Compare L to (HL)
CPMX		117 277	Compare X to (HL)
CPX	data	117 074	Compare X to data
CTC	loc	142 lsb msb	Subroutine call if true carry
CTB	loc	142 lsb msb	Subroutine call if true borrow
CTZ	loc	152 lsb msb	Subroutine call if true zero
CTE	loc	152 lsb msb	Subroutine call if true equal
CTS	loc	162 lsb msb	Subroutine call if true sign
CTL	loc	162 lsb msb	Subroutine call if true less
CTN	loc	162 lsb msb	Subroutine call if true negative
CTP	loc	172 lsb msb	Subroutine call if true parity
DADI	rp,data	rp 110 lsb msb	Double immediate to register add
DACI	rp,data	rp 311 lsb msb	Double immediate to register add with carry
DSUI	rp,data	rp 130 lsb msb	Double immediate to register subtract
DSBI	rp,data	rp 331 lsb msb	Double immediate to register subtract with borrow
DNDI	rp,data	rp 140 lsb msb	Double immediate to register and
DXRI	rp,data	rp 150 lsb msb	Double immediate to register exclusive or
DORI	rp,data	rp 160 lsb msb	Double immediate to register or
DCPI	rp,data	rp 170 lsb msb	Double immediate to register compare
DADM	rp	rp 013	Double memory to register add
DACM	rp	rp 310	Double memory to register

			add with carry
DSUM	rp	rp 033	Double memory to register subtract
DSBM	rp	rp 330	Double memory to register subtract with borrow
DNDM	rp	rp 043	Double memory to register and
DXRM	rp	rp 053	Double memory to register exclusive or
DORM	rp	rp 063	Double memory to register or
DCPM	rp	rp 073	Double memory to register compare
DADP	rp,loc	rp+1 013 lsb	Double paged to register add
DACP	rp,loc	rp+1 310 lsb	Double paged to register add with carry
DSUP	rp,loc	rp+1 033 lsb	Double paged to register subtract
DSBP	rp,loc	rp+1 330 lsb	Double paged to register subtract with borrow
DNDP	rp,loc	rp+1 043 lsb	Double paged to register and
DXRP	rp,loc	rp+1 053 lsb	Double paged to register exclusive or
DORP	rp,loc	rp+1 063 lsb	Double paged to register or
DCPP	rp,loc	rp+1 073 lsb	Double paged to register compare
DECI	disp,index	025 lsb ndx	Decrement index
DECI	*disp,index	111 025 lsb msb ndx	Decrement index
DECP	BC	062 035	Decrement BC pair
DECP	BC,2	113 035	Decrement BC pair by 2
DECP	BC,A	062 037	Decrement BC pair by A
DECP	DE	174 035	Decrement DE pair
DECP	DE,2	115 035	Decrement DE pair by 2
DECP	DE,A	174 037	Decrement DE pair by A
DECP	HL	035	Decrement HL pair
DECP	HL,2	117 035	Decrement HL pair by 2
DECP	HL,A	037	Decrement HL pair by A
DECP	XA	022 035	Decrement XA pair
DECP	XA,2	111 035	Decrement XA pair by 2
DECP	XA,A	022 037	Decrement XA pair by A
DFAC		111 041	Decimal field add with

			carry
DFSB		062 041	Decimal field subtract with borrow
DI		040	Disable interrupts
DIDIV		111 031	Double integer divide
DL	BC,BC	062 047	Double load BC from (BC)
DL	BC,DE	113 047	Double load BC from (DE)
DL	BC,HL	111 047	Double load BC from (HL)
DL	DE,BC	174 047	Double load DE from (BC)
DL	DE,DE	115 047	Double load DE from (DE)
DL	DE,HL	047	Double load DE from (HL)
DL	HL,BC	176 047	Double load HL from (BC)
DL	HL,DE	117 047	Double load HL from (DE)
DL	HL,HL	057	Double load HL from (HL)
DMAD	rp	rp+1 110	Double register to memory add
DMAC	rp	rp+1 311	Double register to memory add with carry
DMSU	rp	rp+1 130	Double register to memory subtract
DMSB	rp	rp+1 331	Double register to memory subtract with borrow
DMND	rp	rp+1 140	Double register to memory and
DMXR	rp	rp+1 150	Double register to memory exclusive or
DMOR	rp	rp+1 160	Double register to memory or
DPL	BC,loc	111 124 lsb	Double paged load BC
DPL	DE,loc	113 144 lsb	Double paged load DE
DPL	HL,loc	115 164 lsb	Double paged load HL
DPLR	BC,loc	062 114 lsb	Double paged load reversed BC
DPLR	DE,loc	174 134 lsb	Double paged load reversed DE
DPLR	HL,loc	176 154 lsb	Double paged load reversed HL
DPS	BC,loc	111 126 lsb	Double paged store BC
DPS	DE,loc	113 146 lsb	Double paged store DE
DPS	HL,loc	115 166 lsb	Double paged store HL
DPSR	BC,loc	062 116 lsb	Double paged store

					reversed BC
DPSR	DE,loc	174	136	lsb	Double paged store reversed DE
DPSR	HL,loc	176	156	lsb	Double paged store reversed HL
DS	BC,DE	113	027		Double store BC into (DE)
DS	BC,HL	111	027		Double store BC into (HL)
DS	DE,BC	174	027		Double store DE into (BC)
DS	DE,HL	027			Double store DE into (HL)
DS	HL,BC	176	027		Double store HL into (BC)
DS	HL,DE	117	027		Double store HL into (DE)
EI		050			Enable interrupts
EJMP	loc	111	050	lsb msb	Enable interrupts and jump
EUR		062	050		Enable interrupts and user return
EX	ADR	121			Output address from A
EX	BEEP	151			Beep
EX	BSP	167			Backspace tape
EX	CLICK	153			Click
EX	COM1	131			External command 1 from A
EX	COM2	133			External command 2 from A
EX	COM3	135			External command 3 from A
EX	COM4	137			External command 4 from A
EX	DATA	125			Select data mode
EX	DECK1	155			Select cassette deck 1
EX	DECK2	157			Select cassette deck 2
EX	RBK	161			Read block
EX	REWIND	175			Rewind cassette deck
EX	SB	173			Slew backward (cassette)
EX	SF	171			Slew forward (cassette)
EX	STATUS	123			Sense status
EX	TSTOP	177			Stop cassette tape
EX	WBK	163			Write block
EX	WRITE	127			Write data from A
EXA	ADR	121			Output address from A
EXA	COM1	131			External command 1 from A
EXA	COM2	133			External command 2 from A
EXA	COM3	135			External command 3 from A
EXA	COM4	137			External command 4 from A
EXA	WRITE	127			Write data from A
EXB	ADR	111	121		Output address from B
EXB	COM1	111	131		External command 1 from B

EXB	COM2	111 133	External command 2 from B
EXB	COM3	111 135	External command 3 from B
EXB	COM4	111 137	External command 4 from B
EXB	WRITE	111 127	Write data from B
EXC	ADR	062 121	Output address from C
EXC	COM1	062 131	External command 1 from C
EXC	COM2	062 133	External command 2 from C
EXC	COM3	062 135	External command 3 from C
EXC	COM4	062 137	External command 4 from C
EXC	WRITE	062 127	Write data from C
EXD	ADR	113 121	Output address from D
EXD	COM1	113 131	External command 1 from D
EXD	COM2	113 133	External command 2 from D
EXD	COM3	113 135	External command 3 from D
EXD	COM4	113 137	External command 4 from D
EXD	WRITE	113 127	Write data from D
EXE	ADR	174 121	Output address from E
EXE	COM1	174 131	External command 1 from E
EXE	COM2	174 133	External command 2 from E
EXE	COM3	174 135	External command 3 from E
EXE	COM4	174 137	External command 4 from E
EXE	WRITE	174 127	Write data from E
EXH	ADR	115 121	Output address from H
EXH	COM1	115 131	External command 1 from H
EXH	COM2	115 133	External command 2 from H
EXH	COM3	115 135	External command 3 from H
EXH	COM4	115 137	External command 4 from H
EXH	WRITE	115 127	Write data from H
EXL	ADR	176 121	Output address from L
EXL	COM1	176 131	External command 1 from L
EXL	COM2	176 133	External command 2 from L
EXL	COM3	176 135	External command 3 from L
EXL	COM4	176 137	External command 4 from L
EXL	WRITE	176 127	Write data from L
EXX	ADR	117 121	Output address from X
EXX	COM1	117 131	External command 1 from X
EXX	COM2	117 133	External command 2 from X
EXX	COM3	117 135	External command 3 from X
EXX	COM4	117 137	External command 4 from X
EXX	WRITE	117 127	Write data from X
HALT		377	Halt

```
IDIV                    062 031             Integer divide
IMULT                   111 011             Integer multiply

IN                      101                 Input to A
INA                     101                 Input to A
INB                     111 101             Input to B
INC                     062 101             Input to C

INCI  disp,index        005 lsb ndx         Increment index
INCI  *disp,index       111 005 lsb msb ndx Increment index

INCP  BC                062 015             Increment BC pair
INCP  BC,2              113 015             Increment BC pair by 2
INCP  BC,A              062 017             Increment BC pair by A
INCP  DE                174 015             Increment DE pair
INCP  DE,2              115 015             Increment DE pair by 2
INCP  DE,A              174 017             Increment DE pair by A
INCP  HL                015                 Increment HL pair
INCP  HL,2              117 015             Increment HL pair by 2
INCP  HL,A              017                 Increment HL pair by A
INCP  XA                022 015             Increment XA pair
INCP  XA,2              111 015             Increment XA pair by 2
INCP  XA,A              022 017             Increment XA pair by A

IND                     113 101             Input to D
INE                     174 101             Input to E

INFO                    111 010             System information

INH                     115 101             Input to H
INL                     176 101             Input to L
INPUT                   101                 Input to A
INX                     117 101             Input to X

JFC   loc               100 lsb msb         Jump if false carry
JFB   loc               100 lsb msb         Jump if false borrow
JFZ   loc               110 lsb msb         Jump if false zero
JFE   loc               110 lsb msb         Jump if false equal
JFS   loc               120 lsb msb         Jump if false sign
JFL   loc               120 lsb msb         Jump if false less
JFN   loc               120 lsb msb         Jump if false negative
JFP   loc               130 lsb msb         Jump if false parity

JMP   loc               104 lsb msb         Jump to location

JTC   loc               140 lsb msb         Jump if true carry
JTB   loc               140 lsb msb         Jump if true borrow
```

JTZ	loc	150 lsb msb	Jump if true zero
JTE	loc	150 lsb msb	Jump if true equal
JTS	loc	160 lsb msb	Jump if true sign
JTL	loc	160 lsb msb	Jump if true less
JTN	loc	160 lsb msb	Jump if true negative
JTP	loc	170 lsb msb	Jump if true parity
JUMP	loc	104 lsb msb	Jump to location
LA	data	006 vvv	Load A with data
LAA		300	Load A from A
LAB		301	Load A from B
LAC		302	Load A from C
LAD		303	Load A from D
LAE		304	Load A from E
LAH		305	Load A from H
LAL		306	Load A from L
LAM		307	Load A from (HL)
LAM	BC	062 307	Load A from (BC)
LAM	DE	174 307	Load A from (DE)
LAM	HL	307	Load A from (HL)
LAM	XA	022 307	Load A from (XA)
LB	data	016 vvv	Load B with data
LBA		310	Load B from A
LBB		311	Load B from B
LBC		312	Load B from C
LBD		313	Load B from D
LBE		314	Load B from E
LBH		315	Load B from H
LBL		316	Load B from L
LBM		317	Load B from (HL)
LBM	BC	062 317	Load B from (BC)
LBM	DE	174 317	Load B from (DE)
LBM	HL	317	Load B from (HL)
LBM	XA	022 317	Load B from (XA)
LC	data	026 vvv	Load C with data
LCA		320	Load C from A
LCB		321	Load C from B
LCC		322	Load C from C
LCD		323	Load C from D
LCE		324	Load C from E
LCH		325	Load C from H
LCL		326	Load C from L
LCM		327	Load C from (HL)
LCM	BC	062 327	Load C from (BC)
LCM	DE	174 327	Load C from (DE)

```
LCM   HL              327                    Load C from (HL)
LCM   XA              022 327                Load C from (XA)

LD    data            036 vvv                Load D with data
LDA                   330                    Load D from A
LDB                   331                    Load D from B
LDC                   332                    Load D from C
LDD                   333                    Load D from D
LDE                   334                    Load D from E
LDH                   335                    Load D from H
LDL                   336                    Load D from L
LDM                   337                    Load D from (HL)
LDM   BC              062 337                Load D from (BC)
LDM   DE              174 337                Load D from (DE)
LDM   HL              337                    Load D from (HL)
LDM   XA              022 337                Load D from (XA)

LE    data            046 vvv                Load E with data
LEA                   340                    Load E from A
LEB                   341                    Load E from B
LEC                   342                    Load E from C
LED                   343                    Load E from D
LEE                   344                    Load E from E
LEH                   345                    Load E from H
LEL                   346                    Load E from L
LEM                   347                    Load E from (HL)
LEM   BC              062 347                Load E from (BC)
LEM   DE              174 347                Load E from (DE)
LEM   HL              347                    Load E from (HL)
LEM   XA              022 347                Load E from (XA)

LFID  BC,disp,index   062 025 lsb ndx        Load BC from index
                                             decremental
LFID  BC,*disp,index  113 025 lsb msb ndx    Load BC from index
                                             decremental
LFID  DE,disp,index   174 025 lsb ndx        Load DE from index
                                             decremental
LFID  DE,*disp,index  115 025 lsb msb ndx    Load DE from index
                                             decremental
LFID  HL,disp,index   176 025 lsb ndx        Load HL from index
                                             decremental
LFID  HL,*disp,index  117 025 lsb msb ndx    Load HL from index
                                             decremental

LFII  BC,disp,index   062 005 lsb ndx        Load BC from index
                                             incremental
LFII  BC,*disp,index  113 005 lsb msb ndx    Load BC from index
                                             incremental
```

LFII	DE,disp,index	174	005	lsb	ndx		Load DE from index incremental
LFII	DE,*disp,index	115	005	lsb	msb	ndx	Load DE from index incremental
LFII	HL,disp,index	176	005	lsb	ndx		Load HL from index incremental
LFII	HL,*disp,index	117	005	lsb	msb	ndx	Load HL from index incremental
LH	data	056	vvv				Load H with data
LHA		350					Load H from A
LHB		351					Load H from B
LHC		352					Load H from C
LHD		353					Load H from D
LHE		354					Load H from E
LHH		355					Load H from H
LHL		356					Load H from L
LHM		357					Load H from (HL)
LHM	BC	062	357				Load H from (BC)
LHM	DE	174	357				Load H from (DE)
LHM	HL	357					Load H from (HL)
LHM	XA	022	357				Load H from (XA)
LL	data	066	vvv				Load L with data
LLA		360					Load L from A
LLB		361					Load L from B
LLC		362					Load L from C
LLD		363					Load L from D
LLDEL		111	051				Doubly linked list delete
LLE		364					Load L from E
LLH		365					Load L from H
LLINS		062	051				Doubly linked list insert
LLL		366					Load L from L
LLM		367					Load L from (HL)
LLM	BC	062	367				Load L from (BC)
LLM	DE	174	367				Load L from (DE)
LLM	HL	367					Load L from (HL)
LLM	XA	022	367				Load L from (XA)
LMA		370					Load (HL) from A
LMA	BC	062	370				Load (BC) from A
LMA	DE	174	370				Load (DE) from A
LMA	HL	370					Load (HL) from A
LMA	XA	022	370				Load (XA) from A

LMB		371	Load (HL) from B
LMB	BC	062 371	Load (BC) from B
LMB	DE	174 371	Load (DE) from B
LMB	HL	371	Load (HL) from B
LMB	XA	022 371	Load (XA) from B
LMC		372	Load (HL) from C
LMC	BC	062 372	Load (BC) from C
LMC	DE	174 372	Load (DE) from C
LMC	HL	372	Load (HL) from C
LMC	XA	022 372	Load (XA) from C
LMD		373	Load (HL) from D
LMD	BC	062 373	Load (BC) from D
LMD	DE	174 373	Load (DE) from D
LMD	HL	373	Load (HL) from D
LMD	XA	022 373	Load (XA) from D
LME		374	Load (HL) from E
LME	BC	062 374	Load (BC) from E
LME	DE	174 374	Load (DE) from E
LME	HL	374	Load (HL) from E
LME	XA	022 374	Load (XA) from E
LMH		375	Load (HL) from H
LMH	BC	062 375	Load (BC) from H
LMH	DE	174 375	Load (DE) from H
LMH	HL	375	Load (HL) from H
LMH	XA	022 375	Load (XA) from H
LML		376	Load (HL) from L
LML	BC	062 376	Load (BC) from L
LML	DE	174 376	Load (DE) from L
LML	HL	376	Load (HL) from L
LML	XA	022 376	Load (XA) from L
LX	data	076 vvv	Load X with data
MIN		111 061	Multiple input
MOUT		111 071	Multiple output
ND	data	044 vvv	AND data to A
NDA		240	AND A to A
NDA	data	044 vvv	AND data to A
NDAA		240	AND A to A
NDAB		111 240	AND A to B
NDAC		062 240	AND A to C
NDAD		113 240	AND A to D
NDAE		174 240	AND A to E
NDAH		115 240	AND A to H
NDAL		176 240	AND A to L
NDAX		117 240	AND A to X

NDB		241	AND B to A
NDB	data	111 044 vvv	AND data to B
NDBA		241	AND B to A
NDBB		111 241	AND B to B
NDBC		062 241	AND B to C
NDBD		113 241	AND B to D
NDBE		174 241	AND B to E
NDBH		115 241	AND B to H
NDBL		176 241	AND B to L
NDBX		117 241	AND B to X
NDC		242	AND C to A
NDC	data	062 044 vvv	AND data to C
NDCA		242	AND C to A
NDCB		111 242	AND C to B
NDCC		062 242	AND C to C
NDCD		113 242	AND C to D
NDCE		174 242	AND C to E
NDCH		115 242	AND C to H
NDCL		176 242	AND C to L
NDCX		117 242	AND C to X
NDD		243	AND D to A
NDD	data	113 044 vvv	AND data to D
NDDA		243	AND D to A
NDDB		111 243	AND D to B
NDDC		062 243	AND D to C
NDDD		113 243	AND D to D
NDDE		174 243	AND D to E
NDDH		115 243	AND D to H
NDDL		176 243	AND D to L
NDDX		117 243	AND D to X
NDE		244	AND E to A
NDE	data	174 044 vvv	AND data to E
NDEA		244	AND E to A
NDEB		111 244	AND E to B
NDEC		062 244	AND E to C
NDED		113 244	AND E to D
NDEE		174 244	AND E to E
NDEH		115 244	AND E to H
NDEL		176 244	AND E to L
NDEX		117 244	AND E to X
NDH		245	AND H to A
NDH	data	115 044 vvv	AND data to H
NDHA		245	AND H to A
NDHB		111 245	AND H to B

```
NDHC                    062 245             AND H to C
NDHD                    113 245             AND H to D
NDHE                    174 245             AND H to E
NDHH                    115 245             AND H to H
NDHL                    176 245             AND H to L
NDHX                    117 245             AND H to X

NDL                     246                 AND L to A
NDL     data            176 044 vvv         AND data to L
NDLA                    246                 AND L to A
NDLB                    111 246             AND L to B
NDLC                    062 246             AND L to C
NDLD                    113 246             AND L to D
NDLE                    174 246             AND L to E
NDLH                    115 246             AND L to H
NDLL                    176 246             AND L to L
NDLX                    117 246             AND L to X

NDM                     247                 AND (HL) to A
NDMA                    247                 AND (HL) to A
NDMB                    111 247             AND (HL) to B
NDMC                    062 247             AND (HL) to C
NDMD                    113 247             AND (HL) to D
NDME                    174 247             AND (HL) to E
NDMH                    115 247             AND (HL) to H
NDML                    176 247             AND (HL) to L
NDMX                    117 247             AND (HL) to X

NDX     data            117 044 vvv         AND data to X

NOJ     loc             045 lsb msb         No jump (3 byte NOP)
NOP                     300                 No operation

OR      data            064 vvv             Inclusive OR data to A
ORA                     260                 Inclusive OR A to A
ORA     data            064 vvv             Inclusive OR data to A
ORAA                    260                 Inclusive OR A to A
ORAB                    111 260             Inclusive OR A to B
ORAC                    062 260             Inclusive OR A to C
ORAD                    113 260             Inclusive OR A to D
ORAE                    174 260             Inclusive OR A to E
ORAH                    115 260             Inclusive OR A to H
ORAL                    176 260             Inclusive OR A to L
ORAX                    117 260             Inclusive OR A to X

ORB                     261                 Inclusive OR B to A
ORB     data            111 064 vvv         Inclusive OR data to B
ORBA                    261                 Inclusive OR B to A
```

ORBB		111 261		Inclusive OR B to B
ORBC		062 261		Inclusive OR B to C
ORBD		113 261		Inclusive OR B to D
ORBE		174 261		Inclusive OR B to E
ORBH		115 261		Inclusive OR B to H
ORBL		176 261		Inclusive OR B to L
ORBX		117 261		Inclusive OR B to X
ORC		262		Inclusive OR C to A
ORC	data	062 064	vvv	Inclusive OR data to C
ORCA		262		Inclusive OR C to A
ORCB		111 262		Inclusive OR C to B
ORCC		062 262		Inclusive OR C to C
ORCD		113 262		Inclusive OR C to D
ORCE		174 262		Inclusive OR C to E
ORCH		115 262		Inclusive OR C to H
ORCL		176 262		Inclusive OR C to L
ORCX		117 262		Inclusive OR C to X
ORD		263		Inclusive OR D to A
ORD	data	113 064	vvv	Inclusive OR data to D
ORDA		263		Inclusive OR D to A
ORDB		111 263		Inclusive OR D to B
ORDC		062 263		Inclusive OR D to C
ORDD		113 263		Inclusive OR D to D
ORDE		174 263		Inclusive OR D to E
ORDH		115 263		Inclusive OR D to H
ORDL		176 263		Inclusive OR D to L
ORDX		117 263		Inclusive OR D to X
ORE		264		Inclusive OR E to A
ORE	data	174 064	vvv	Inclusive OR data to E
OREA		264		Inclusive OR E to A
OREB		111 264		Inclusive OR E to B
OREC		062 264		Inclusive OR E to C
ORED		113 264		Inclusive OR E to D
OREE		174 264		Inclusive OR E to E
OREH		115 264		Inclusive OR E to H
OREL		176 264		Inclusive OR E to L
OREX		117 264		Inclusive OR E to X
ORH		265		Inclusive OR H to A
ORH	data	115 064	vvv	Inclusive OR data to H
ORHA		265		Inclusive OR H to A
ORHB		111 265		Inclusive OR H to B
ORHC		062 265		Inclusive OR H to C
ORHD		113 265		Inclusive OR H to D
ORHE		174 265		Inclusive OR H to E

ORHH		115 265	Inclusive OR H to H
ORHL		176 265	Inclusive OR H to L
ORHX		117 265	Inclusive OR H to X
ORL		266	Inclusive OR L to A
ORL	data	176 064 vvv	Inclusive OR data to L
ORLA		266	Inclusive OR L to A
ORLB		111 266	Inclusive OR L to B
ORLC		062 266	Inclusive OR L to C
ORLD		113 266	Inclusive OR L to D
ORLE		174 266	Inclusive OR L to E
ORLH		115 266	Inclusive OR L to H
ORLL		176 266	Inclusive OR L to L
ORLX		117 266	Inclusive OR L to X
ORM		267	Inclusive OR (HL) to A
ORMA		267	Inclusive OR (HL) to A
ORMB		111 267	Inclusive OR (HL) to B
ORMC		062 267	Inclusive OR (HL) to C
ORMD		113 267	Inclusive OR (HL) to D
ORME		174 267	Inclusive OR (HL) to E
ORMH		115 267	Inclusive OR (HL) to H
ORML		176 267	Inclusive OR (HL) to L
ORMX		117 267	Inclusive OR (HL) to X
ORX	data	117 064 vvv	Inclusive OR data to X
PAD	r,loc	r 106 lsb	Single paged to register add
PAC	r,loc	r 112 lsb	Single paged to register add with carry
PSU	r,loc	r 122 lsb	Single paged to register subtract
PSB	r,loc	r 132 lsb	Single paged to register subtract with borrow
PND	r,loc	r 142 lsb	Single paged to register and
PXR	r,loc	r 152 lsb	Single paged to register exclusive or
POR	r,loc	r 162 lsb	Single paged to register or
PCP	r,loc	r 172 lsb	Single paged to register compare
PIN		103	Parity checking input to A
PINA		103	Parity checking input to A

PINB		111	103	Parity checking input to B
PINC		062	103	Parity checking input to C
PIND		113	103	Parity checking input to D
PINE		174	103	Parity checking input to E
PINH		115	103	Parity checking input to H
PINL		176	103	Parity checking input to L
PINX		117	103	Parity checking input to X
PL	A,loc	105	lsb	Paged load A
PL	B,loc	114	lsb	Paged load B
PL	C,loc	124	lsb	Paged load C
PL	D,loc	134	lsb	Paged load D
PL	E,loc	144	lsb	Paged load E
PL	H,loc	154	lsb	Paged load H
PL	L,loc	164	lsb	Paged load L
POP		060		Pop value from stack into HL
POP	BC	062	060	Pop value from stack into BC
POP	DE	174	060	Pop value from stack into DE
POP	HL	060		Pop value from stack into HL
POP	XA	022	060	Pop value from stack into XA
PS	A,loc	107	lsb	Paged store A
PS	B,loc	116	lsb	Paged store B
PS	C,loc	126	lsb	Paged store C
PS	D,loc	136	lsb	Paged store D
PS	E,loc	146	lsb	Paged store E
PS	H,loc	156	lsb	Paged store H
PS	L,loc	166	lsb	Paged store L
PUSH		070		Push HL onto stack
PUSH	data	051	lsb msb	Push data onto stack
PUSH	BC	062	070	Push BC onto stack
PUSH	DE	174	070	Push DE onto stack
PUSH	HL	070		Push HL onto stack
PUSH	XA	022	070	Push XA onto stack

REGL	111 055	Register load
REGS	055	Register save
RET	007	Subroutine return
RETURN	007	Subroutine return
RFC	003	Subroutine return if false carry
RFB	003	Subroutine return if false borrow
RFZ	013	Subroutine return if false zero
RFE	013	Subroutine return if false equal
RFS	023	Subroutine return if false sign
RFL	023	Subroutine return if false less
RFN	023	Subroutine return if false negative
RFP	033	Subroutine return if false parity
RTC	043	Subroutine return if true carry
RTB	043	Subroutine return if true borrow
RTZ	053	Subroutine return if true zero
RTE	053	Subroutine return if true equal
RTS	063	Subroutine return if true sign
RTL	063	Subroutine return if true less
RTN	063	Subroutine return if true negative
RTP	073	Subroutine return if true parity
SB data	034 vvv	Subtract with borrow data from A
SBA	230	Subtract with borrow A from A
SBA data	034 vvv	Subtract with borrow data from A
SBAA	230	Subtract with borrow A

			from A
SBAB		111 230	Subtract with borrow A from B
SBAC		062 230	Subtract with borrow A from C
SBAD		113 230	Subtract with borrow A from D
SBAE		174 230	Subtract with borrow A from E
SBAH		115 230	Subtract with borrow A from H
SBAL		176 230	Subtract with borrow A from L
SBAX		117 230	Subtract with borrow A from X
SBB		231	Subtract with borrow B from A
SBB	data	111 034 vvv	Subtract with borrow data from B
SBBA		231	Subtract with borrow B from A
SBBB		111 231	Subtract with borrow B from B
SBBC		062 231	Subtract with borrow B from C
SBBD		113 231	Subtract with borrow B from D
SBBE		174 231	Subtract with borrow B from E
SBBH		115 231	Subtract with borrow B from H
SBBL		176 231	Subtract with borrow B from L
SBBX		117 231	Subtract with borrow B from X
SBC		232	Subtract with borrow C from A
SBC	data	062 034 vvv	Subtract with borrow data from C
SBCA		232	Subtract with borrow C from A
SBCB		111 232	Subtract with borrow C from B
SBCC		062 232	Subtract with borrow C from C
SBCD		113 232	Subtract with borrow C

				from D
SBCE		174	232	Subtract with borrow C from E
SBCH		115	232	Subtract with borrow C from H
SBCL		176	232	Subtract with borrow C from L
SBCX		117	232	Subtract with borrow C from X
SBD		233		Subtract with borrow D from A
SBD	data	113	034 vvv	Subtract with borrow data from D
SBDA		233		Subtract with borrow D from A
SBDB		111	233	Subtract with borrow D from B
SBDC		062	233	Subtract with borrow D from C
SBDD		113	233	Subtract with borrow D from D
SBDE		174	233	Subtract with borrow D from E
SBDH		115	233	Subtract with borrow D from H
SBDL		176	233	Subtract with borrow D from L
SBDX		117	233	Subtract with borrow D from X
SBE		234		Subtract with borrow E from A
SBE	data	174	034 vvv	Subtract with borrow data from E
SBEA		234		Subtract with borrow E from A
SBEB		111	234	Subtract with borrow E from B
SBEC		062	234	Subtract with borrow E from C
SBED		113	234	Subtract with borrow E from D
SBEE		174	234	Subtract with borrow E from E
SBEH		115	234	Subtract with borrow E from H
SBEL		176	234	Subtract with borrow E

					from L
SBEX		117	234		Subtract with borrow E from X
SBH		235			Subtract with borrow H from A
SBH	data	115	034	vvv	Subtract with borrow data from H
SBHA		235			Subtract with borrow H from A
SBHB		111	235		Subtract with borrow H from B
SBHC		062	235		Subtract with borrow H from C
SBHD		113	235		Subtract with borrow H from D
SBHE		174	235		Subtract with borrow H from E
SBHH		115	235		Subtract with borrow H from H
SBHL		176	235		Subtract with borrow H from L
SBHX		117	235		Subtract with borrow H from X
SBL		236			Subtract with borrow L from A
SBL	data	176	034	vvv	Subtract with borrow data from L
SBLA		236			Subtract with borrow L from A
SBLB		111	236		Subtract with borrow L from B
SBLC		062	236		Subtract with borrow L from C
SBLD		113	236		Subtract with borrow L from D
SBLE		174	236		Subtract with borrow L from E
SBLH		115	236		Subtract with borrow L from H
SBLL		176	236		Subtract with borrow L from L
SBLX		117	236		Subtract with borrow L from X
SBM		237			Subtract with borrow (HL) from A

SBMA	237		Subtract with borrow (HL) from A
SBMB	111 237		Subtract with borrow (HL) from B
SBMC	062 237		Subtract with borrow (HL) from C
SBMD	113 237		Subtract with borrow (HL) from D
SBME	174 237		Subtract with borrow (HL) from E
SBMH	115 237		Subtract with borrow (HL) from H
SBML	176 237		Subtract with borrow (HL) from L
SBMX	117 237		Subtract with borrow (HL) from X
SBX data	117 034	vvv	Subtract with borrow data from X
SC	067		System call
SLC	002		Shift A left circular
SLCA	002		Shift A left circular
SLCB	111 002		Shift B left circular
SLCC	062 002		Shift C left circular
SLCD	113 002		Shift D left circular
SLCE	174 002		Shift E left circular
SLCH	115 002		Shift H left circular
SLCL	176 002		Shift L left circular
SLCX	117 002		Shift X left circular
SRC	012		Shift A right circular
SRCA	012		Shift A right circular
SRCB	111 012		Shift B right circular
SRCC	062 012		Shift C right circular
SRCD	113 012		Shift D right circular
SRCE	174 012		Shift E right circular
SRCH	115 012		Shift H right circular
SRCL	176 012		Shift L right circular
SRCX	117 012		Shift X right circular
SRE	032		Shift A right extended
SREA	032		Shift A right extended
SREB	111 032		Shift B right extended
SREC	062 032		Shift C right extended
SRED	113 032		Shift D right extended
SREE	174 032		Shift E right extended

SREH	115 032		Shift H right extended
SREL	176 032		Shift L right extended
SREX	117 032		Shift X right extended
STKL	111 065		Stack load
STKS	065		Stack save
STL	077		Sector table load
STLO	022 077		Sector table load starting at offset A
STLOA	022 077		Sector table load starting at offset A
STLOB	111 077		Sector table load starting at offset B
STLOC	062 077		Sector table load starting at offset C
STLOD	113 077		Sector table load starting at offset D
STLOE	174 077		Sector table load starting at offset E
STLOH	115 077		Sector table load starting at offset H
STLOL	176 077		Sector table load starting at offset L
STLOX	117 077		Sector table load starting at offset X
SU data	024 vvv		Subtract data from A
SUA	220		Subtract A from A
SUA data	024 vvv		Subtract data from A
SUAA	220		Subtract A from A
SUAB	111 220		Subtract A from B
SUAC	062 220		Subtract A from C
SUAD	113 220		Subtract A from D
SUAE	174 220		Subtract A from E
SUAH	115 220		Subtract A from H
SUAL	176 220		Subtract A from L
SUAX	117 220		Subtract A from X
SUB	221		Subtract B from A
SUB data	111 024	vvv	Subtract data from B
SUBA	221		Subtract B from A
SUBB	111 221		Subtract B from B
SUBC	062 221		Subtract B from C
SUBD	113 221		Subtract B from D
SUBE	174 221		Subtract B from E
SUBH	115 221		Subtract B from H

```
SUBL                    176 221             Subtract B from L
SUBX                    117 221             Subtract B from X

SUC                     222                 Subtract C from A
SUC     data            062 024 vvv         Subtract data from C
SUCA                    222                 Subtract C from A
SUCB                    111 222             Subtract C from B
SUCC                    062 222             Subtract C from C
SUCD                    113 222             Subtract C from D
SUCE                    174 222             Subtract C from E
SUCH                    115 222             Subtract C from H
SUCL                    176 222             Subtract C from L
SUCX                    117 222             Subtract C from X

SUD                     223                 Subtract D from A
SUD     data            113 024 vvv         Subtract data from D
SUDA                    223                 Subtract D from A
SUDB                    111 223             Subtract D from B
SUDC                    062 223             Subtract D from C
SUDD                    113 223             Subtract D from D
SUDE                    174 223             Subtract D from E
SUDH                    115 223             Subtract D from H
SUDL                    176 223             Subtract D from L
SUDX                    117 223             Subtract D from X

SUE                     224                 Subtract E from A
SUE     data            174 024 vvv         Subtract data from E
SUEA                    224                 Subtract E from A
SUEB                    111 224             Subtract E from B
SUEC                    062 224             Subtract E from C
SUED                    113 224             Subtract E from D
SUEE                    174 224             Subtract E from E
SUEH                    115 224             Subtract E from H
SUEL                    176 224             Subtract E from L
SUEX                    117 224             Subtract E from X

SUH                     225                 Subtract H from A
SUH     data            115 024 vvv         Subtract data from H
SUHA                    225                 Subtract H from A
SUHB                    111 225             Subtract H from B
SUHC                    062 225             Subtract H from C
SUHD                    113 225             Subtract H from D
SUHE                    174 225             Subtract H from E
SUHH                    115 225             Subtract H from H
SUHL                    176 225             Subtract H from L
SUHX                    117 225             Subtract H from X

SUL                     226                 Subtract L from A
```

SUL	data	176	024	vvv	Subtract data from L
SULA		226			Subtract L from A
SULB		111	226		Subtract L from B
SULC		062	226		Subtract L from C
SULD		113	226		Subtract L from D
SULE		174	226		Subtract L from E
SULH		115	226		Subtract L from H
SULL		176	226		Subtract L from L
SULX		117	226		Subtract L from X
SUM		227			Subtract (HL) from A
SUMA		227			Subtract (HL) from A
SUMB		111	227		Subtract (HL) from B
SUMC		062	227		Subtract (HL) from C
SUMD		113	227		Subtract (HL) from D
SUME		174	227		Subtract (HL) from E
SUMH		115	227		Subtract (HL) from H
SUML		176	227		Subtract (HL) from L
SUMX		117	227		Subtract (HL) from X
SUX	data	117	024	vvv	Subtract data from X
SYNC		010			Generate sync pulse
UR		111	102		User return
XR	data	054	vvv		Exclusive OR data to A
XRA		250			Exclusive OR A to A
XRA	data	054	vvv		Exclusive OR data to A
XRAA		250			Exclusive OR A to A
XRAB		111	250		Exclusive OR A to B
XRAC		062	250		Exclusive OR A to C
XRAD		113	250		Exclusive OR A to D
XRAE		174	250		Exclusive OR A to E
XRAH		115	250		Exclusive OR A to H
XRAL		176	250		Exclusive OR A to L
XRAX		117	250		Exclusive OR A to X
XRB		251			Exclusive OR B to A
XRB	data	111	054	vvv	Exclusive OR data to B
XRBA		251			Exclusive OR B to A
XRBB		111	251		Exclusive OR B to B
XRBC		062	251		Exclusive OR B to C
XRBD		113	251		Exclusive OR B to D
XRBE		174	251		Exclusive OR B to E
XRBH		115	251		Exclusive OR B to H
XRBL		176	251		Exclusive OR B to L
XRBX		117	251		Exclusive OR B to X

XRC		252		Exclusive OR C to A
XRC	data	062	054 vvv	Exclusive OR data to C
XRCA		252		Exclusive OR C to A
XRCB		111	252	Exclusive OR C to B
XRCC		062	252	Exclusive OR C to C
XRCD		113	252	Exclusive OR C to D
XRCE		174	252	Exclusive OR C to E
XRCH		115	252	Exclusive OR C to H
XRCL		176	252	Exclusive OR C to L
XRCX		117	252	Exclusive OR C to X
XRD		253		Exclusive OR D to A
XRD	data	113	054 vvv	Exclusive OR data to D
XRDA		253		Exclusive OR D to A
XRDB		111	253	Exclusive OR D to B
XRDC		062	253	Exclusive OR D to C
XRDD		113	253	Exclusive OR D to D
XRDE		174	253	Exclusive OR D to E
XRDH		115	253	Exclusive OR D to H
XRDL		176	253	Exclusive OR D to L
XRDX		117	253	Exclusive OR D to X
XRE		254		Exclusive OR E to A
XRE	data	174	054 vvv	Exclusive OR data to E
XREA		254		Exclusive OR E to A
XREB		111	254	Exclusive OR E to B
XREC		062	254	Exclusive OR E to C
XRED		113	254	Exclusive OR E to D
XREE		174	254	Exclusive OR E to E
XREH		115	254	Exclusive OR E to H
XREL		176	254	Exclusive OR E to L
XREX		117	254	Exclusive OR E to X
XRH		255		Exclusive OR H to A
XRH	data	115	054 vvv	Exclusive OR data to H
XRHA		255		Exclusive OR H to A
XRHB		111	255	Exclusive OR H to B
XRHC		062	255	Exclusive OR H to C
XRHD		113	255	Exclusive OR H to D
XRHE		174	255	Exclusive OR H to E
XRHH		115	255	Exclusive OR H to H
XRHL		176	255	Exclusive OR H to L
XRHX		117	255	Exclusive OR H to X

XRL		256			Exclusive OR L to A
XRL	data	176	054	vvv	Exclusive OR data to L
XRLA		256			Exclusive OR L to A
XRLB		111	256		Exclusive OR L to B
XRLC		062	256		Exclusive OR L to C
XRLD		113	256		Exclusive OR L to D
XRLE		174	256		Exclusive OR L to E
XRLH		115	256		Exclusive OR L to H
XRLL		176	256		Exclusive OR L to L
XRLX		117	256		Exclusive OR L to X
XRM		257			Exclusive OR (HL) to A
XRMA		257			Exclusive OR (HL) to A
XRMB		111	257		Exclusive OR (HL) to B
XRMC		062	257		Exclusive OR (HL) to C
XRMD		113	257		Exclusive OR (HL) to D
XRME		174	257		Exclusive OR (HL) to E
XRMH		115	257		Exclusive OR (HL) to H
XRML		176	257		Exclusive OR (HL) to L
XRMX		117	257		Exclusive OR (HL) to X
XRX	data	117	054	vvv	Exclusive OR data to X

APPENDIX C. RESERVED MNEMONICS

The mnemonics in the following list are predefined for use in the instruction field as SNAP/3 directives. Macros must be assigned names which do not conflict with these predefined mnemonics.

ALIGN	IFGT	IFZ	MIFGE	ORG	TESTGT	TM
DA	IFLE	INC	MIFGT	PROG	TESTLE	TP
DC	IFLT	LIS	MIFLE	RPT	TESTLT	USE
END	IFNE	LIST	MIFLT	SET	TESTNE	XIF
EQU	IFNG	LOC	MIFNE	SK	TESTNG	
ERR	IFNL	MACRO	MIFNG	SKIP	TESTNL	
IF	IFNSTR	MEND	MIFNL	SNAPOPT	TESTNZ	
IFC	IFNZ	MIF	MIFS	TESTC	TESTS	
IFEQ	IFS	MIFC	MLIB	TESTEQ	TESTZ	
IFGE	IFSTR	MIFEQ	MXIF	TESTGE	TITLE	

The mnemonics in the following list are the Datapoint 2200 instructions, and may not be used as macro names unless the "U" option appears on the SNAP/3 command line.

AC	CFL	JFL	LDA	MLB	ORHA	SLCA
ACA	CFN	JFN	LDB	MLC	ORL	SLN
ACAA	CFP	JFP	LDC	MLD	ORLA	SRC
ACB	CFS	JFS	LDD	MLE	ORM	SRCA
ACBA	CFZ	JFZ	LDE	MLH	ORMA	SRN
ACC	CP	JMP	LDH	MLL	POP	SU
ACCA	CPA	JTB	LDL	MSA	PUSH	SUA
ACD	CPAA	JTC	LDM	MSB	RET	SUAA
ACDA	CPB	JTE	LE	MSC	RETURN	SUB
ACE	CPBA	JTL	LEA	MSD	RFB	SUBA
ACEA	CPC	JTN	LEB	MSE	RFC	SUC
ACH	CPCA	JTP	LEC	MSH	RFE	SUCA
ACHA	CPD	JTS	LED	MSL	RFL	SUD
ACL	CPDA	JTZ	LEE	ND	RFN	SUDA
ACLA	CPE	JUMP	LEH	NDA	RFP	SUE
ACM	CPEA	LA	LEL	NDAA	RFS	SUEA
ACMA	CPH	LAA	LEM	NDB	RFZ	SUH
AD	CPHA	LAB	LH	NDBA	RTB	SUHA
ADA	CPL	LAC	LHA	NDC	RTC	SUL
ADAA	CPLA	LAD	LHB	NDCA	RTE	SULA
ADB	CPM	LAE	LHC	NDD	RTL	SUM
ADBA	CPMA	LAH	LHD	NDDA	RTN	SUMA
ADC	CTB	LAL	LHE	NDE	RTP	SYNC
ADCA	CTC	LAM	LHH	NDEA	RTS	XR
ADD	CTE	LB	LHL	NDH	RTZ	XRA
ADDA	CTL	LBA	LHM	NDHA	SB	XRAA
ADE	CTN	LBB	LL	NDL	SBA	XRB
ADEA	CTP	LBC	LLA	NDLA	SBAA	XRBA
ADH	CTS	LBD	LLB	NDM	SBB	XRC
ADHA	CTZ	LBE	LLC	NDMA	SBBA	XRCA
ADL	DE	LBH	LLD	NOP	SBC	XRD
ADLA	DI	LBL	LLE	OR	SBCA	XRDA
ADM	EI	LBM	LLH	ORA	SBD	XRE
ADMA	EX	LC	LLL	ORAA	SBDA	XREA
ALPHA	EXA	LCA	LLM	ORB	SBE	XRH
BC	HALT	LCB	LMA	ORBA	SBEA	XRHA
BETA	HL	LCC	LMB	ORC	SBH	XRL
CALL	IN	LCD	LMC	ORCA	SBHA	XRLA
CCL	INA	LCE	LMD	ORD	SBL	XRM
CCLA	INPUT	LCH	LME	ORDA	SBLA	XRMA
CFB	JFB	LCL	LMH	ORE	SBM	
CFC	JFC	LCM	LML	OREA	SBMA	
CFE	JFE	LD	MLA	ORH	SLC	

The mnemonics in the following list are the additional Datapoint 5500 instructions, and may not be used as macro names unless the "U" option appears on the SNAP/3 command line.

ACAB	ACHE	ADCL	BCP	CPAL	CPLC	INX
ACAC	ACHH	ADCX	BCV	CPAX	CPLD	LFID
ACAD	ACHL	ADDB	BFAC	CPBB	CPLE	LFII
ACAE	ACHX	ADDC	BFSB	CPBC	CPLH	LX
ACAH	ACLB	ADDD	BFSL	CPBD	CPLL	MIN
ACAL	ACLC	ADDE	BFSR	CPBE	CPLX	MOUT
ACAX	ACLD	ADDH	BP	CPBH	CPMB	NDAB
ACBB	ACLE	ADDL	BRL	CPBL	CPMC	NDAC
ACBC	ACLH	ADDX	BRLA	CPBX	CPMD	NDAD
ACBD	ACLL	ADEB	BRLB	CPCB	CPME	NDAE
ACBE	ACLX	ADEC	BRLC	CPCC	CPMH	NDAH
ACBH	ACMB	ADED	BRLD	CPCD	CPML	NDAL
ACBL	ACMC	ADEE	BRLE	CPCE	CPMX	NDAX
ACBX	ACMD	ADEH	BRLH	CPCH	CPX	NDBB
ACCB	ACME	ADEL	BRLL	CPCL	DECI	NDBC
ACCC	ACMH	ADEX	BRLX	CPCX	DECP	NDBD
ACCD	ACML	ADHB	BT	CPDB	DFAC	NDBE
ACCE	ACMX	ADHC	BTR	CPDC	DFSB	NDBH
ACCH	ACX	ADHD	CCLB	CPDD	DL	NDBL
ACCL	ADAB	ADHE	CCLC	CPDE	DPL	NDBX
ACCX	ADAC	ADHH	CCLD	CPDH	DPS	NDCB
ACDB	ADAD	ADHL	CCLE	CPDL	DS	NDCC
ACDC	ADAE	ADHX	CCLH	CPDX	EJMP	NDCD
ACDD	ADAH	ADLB	CCLL	CPEB	EUR	NDCE
ACDE	ADAL	ADLC	CCS	CPEC	EXB	NDCH
ACDH	ADAX	ADLD	CCSA	CPED	EXC	NDCL
ACDL	ADBB	ADLE	CCSB	CPEE	EXD	NDCX
ACDX	ADBC	ADLH	CCSC	CPEH	EXE	NDDB
ACEB	ADBD	ADLL	CCSD	CPEL	EXH	NDDC
ACEC	ADBE	ADLX	CCSE	CPEX	EXL	NDDD
ACED	ADBH	ADMB	CCSH	CPHB	EXX	NDDE
ACEE	ADBL	ADMC	CCSL	CPHC	INB	NDDH
ACEH	ADBX	ADMD	CCSX	CPHD	INCI	NDDL
ACEL	ADCB	ADME	CPAB	CPHE	INCP	NDDX
ACEX	ADCC	ADMH	CPAC	CPHH	IND	NDEB
ACHB	ADCD	ADML	CPAD	CPHL	INE	NDEC
ACHC	ADCE	ADMX	CPAE	CPHX	INH	NDED
ACHD	ADCH	ADX	CPAH	CPLB	INL	NDEE

NDEH	ORCE	PINC	SBEL	SRED	SUHB	XRCL
NDEL	ORCH	PIND	SBEX	SREE	SUHC	XRCX
NDEX	ORCL	PINE	SBHB	SREH	SUHD	XRDB
NDHB	ORCX	PINH	SBHC	SREL	SUHE	XRDC
NDHC	ORDB	PINL	SBHD	SREX	SUHH	XRDD
NDHD	ORDC	PINX	SBHE	STKL	SUHL	XRDE
NDHE	ORDD	PL	SBHH	STKS	SUHX	XRDH
NDHH	ORDE	PS	SBHL	STL	SULB	XRDL
NDHL	ORDH	REGL	SBHX	SUAB	SULC	XRDX
NDHX	ORDL	REGS	SBLB	SUAC	SULD	XREB
NDLB	ORDX	SBAB	SBLC	SUAD	SULE	XREC
NDLC	OREB	SBAC	SBLD	SUAE	SULH	XRED
NDLD	OREC	SBAD	SBLE	SUAH	SULL	XREE
NDLE	ORED	SBAE	SBLH	SUAL	SULX	XREH
NDLH	OREE	SBAH	SBLL	SUAX	SUMB	XREL
NDLL	OREH	SBAL	SBLX	SUBB	SUMC	XREX
NDLX	OREL	SBAX	SBMB	SUBC	SUMD	XRHB
NDMB	OREX	SBBB	SBMC	SUBD	SUME	XRHC
NDMC	ORHB	SBBC	SBMD	SUBE	SUMH	XRHD
NDMD	ORHC	SBBD	SBME	SUBH	SUML	XRHE
NDME	ORHD	SBBE	SBMH	SUBL	SUMX	XRHH
NDMH	ORHE	SBBH	SBML	SUBX	SUX	XRHL
NDML	ORHH	SBBL	SBMX	SUCB	UR	XRHX
NDMX	ORHL	SBBX	SBX	SUCC	XA	XRLB
NDX	ORHX	SBCB	SC	SUCD	XRAB	XRLC
NOJ	ORLB	SBCC	SLCB	SUCE	XRAC	XRLD
ORAB	ORLC	SBCD	SLCC	SUCH	XRAD	XRLE
ORAC	ORLD	SBCE	SLCD	SUCL	XRAE	XRLH
ORAD	ORLE	SBCH	SLCE	SUCX	XRAH	XRLL
ORAE	ORLH	SBCL	SLCH	SUDB	XRAL	XRLX
ORAH	ORLL	SBCX	SLCL	SUDC	XRAX	XRMB
ORAL	ORLX	SBDB	SLCX	SUDD	XRBB	XRMC
ORAX	ORMB	SBDC	SRCB	SUDE	XRBC	XRMD
ORBB	ORMC	SBDD	SRCC	SUDH	XRBD	XRME
ORBC	ORMD	SBDE	SRCD	SUDL	XRBE	XRMH
ORBD	ORME	SBDH	SRCE	SUDX	XRBH	XRML
ORBE	ORMH	SBDL	SRCH	SUEB	XRBL	XRMX
ORBH	ORML	SBDX	SRCL	SUEC	XRBX	XRX
ORBL	ORMX	SBEB	SRCX	SUED	XRCB	INFO
ORBX	ORX	SBEC	SRE	SUEE	XRCC	
ORCB	PIN	SBED	SREA	SUEH	XRCD	
ORCC	PINA	SBEE	SREB	SUEL	XRCE	
ORCD	PINB	SBEH	SREC	SUEX	XRCH	

The mnemonics in the following list are the additional Datapoint 6600 instructions on the SNAP/3 command line.

BFLRAC	DACM	DMAD	DORM	DXRI	PCP	STLOD
BFLRAD	DACP	DMND	DORP	DXRM	PND	STLOE
BFLRND	DADI	DMOR	DPLR	DXRP	POR	STLOH
BFLROR	DADM	DMSB	DPSR	IDIV	PSB	STLOL
BFLRSB	DADP	DMSU	DSBI	IMULT	PSU	STLOX
BFLRSU	DCPI	DMXR	DSBM	PXR		
BFLRXR	DCPM	DNDI	DSBP	LLDEL	STLO	
COMP	DCPP	DNDM	DSUI	LLINS	STLOA	
COMPS	DIDIV	DNDP	DSUM	PAC	STLOB	
DACI	DMAC	DORI	DSUP	PAD	STLOC	

APPENDIX D. INSTALLATION INSTRUCTIONS

Installation of SNAP/3 requires the following prerequisites:

1. Datapoint processor with 5500 instruction set with Disk Operating System
2. SORT or FASTSORT Utility
3. MIN Utility

The SNAP/3 Macro Assembler is installed by loading the Disk Operating System, inserting the program distribution cassette in the front deck and typing the command:

```
MIN ;AO
```

When loading is complete the following module will have been cataloged on your disk:

SNAP3/CMD	SNAP/3 Assembler

The following programs will be required for library maintenance and for the conversion of relocatabe to executable (absolute) code.

1. LIBSYS Utility
2. LINK Utility (Version 2.1 or later)

APPENDIX E. INCOMPATABILITIES WITH SNAP AND SNAP/2

1. The IFEQ, etc., directives use signed comparisons between their operands; SNAP and SNAP/2 used unsigned comparisons. This may cause different results if two addresses having different high bits are compared. The multiply and divide operators also use signed arithmetic.

2. A reference to a multiply defined label will use the most recent previous definition, or the last definition if the label has not been defined previously. With SNAP and SNAP/2, the last definition was always used during pass 2.

3. Any extra parameters supplied for directives, instructions, or pseudo-instructions will produce an error flag.

4. The ALIGN, TM, and TP directives use the location counter value to determine how much to skip, instead of the address counter. This can only cause different results if the LOC directive is used.

5. Under SNAP and SNAP/2, if a local label in a macro appeared on a line containing a call to a second macro, and there was no label on the prototype header of the second macro, an additional line was created at the beginning of the expansion for the second macro call, and the label therefore became global. This is undesirable, so with SNAP/3 the label remains local, no additional expansion is created, and the label is defined with the expected value before the expanded second macro is processed.

6. The format of the parameters for the MIFEQ, etc., directives was simplified and generalized. Each parameter is an arbitrary string terminated by the first unforced blank or comma not in quotes. Normal substitution will be made for any macro parameter appearing in either string. No special rules apply if either string is null after substitution. These changes may cause different results if the second string contains macro parameters or concatenation characters.

7. The T option will force SNAP/3 to make two passes, and therefore the generated relocatable file will not contain forward reference entries in the external reference table. This option must be used to generate files which are to be

loaded by the DOS relocatable loader (DOS function 15).

8. The LIST directive is processed slightly differently to allow LIST directives to be nested. A LIST x statement will not always turn on the listing control flag if several LIST -x statements have appeared in a row (see section 3.22).

9. LINK version 2.1 or later must be used to link relocatable programs generated by SNAP/3.

10. In SNAP/2 and SNAP/3 the entry point file name defaults to the PROG name, which defaults to the object file name. In SNAP the entry point file name defaults to the source file name.

INDEX

Manual Name_____

Manual Number_____

READER'S COMMENTS

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Name_____ Date_____

Organization_____

Street_____

City_____ State_____ Zip Code_____

All comments and suggestions become the property of Datapoint.

Fold Here

Fold Here and Staple

FIRST CLASS
Permit
5774
San Antonio
Texas

BUSINESS REPLY MAIL
No Postage Necessary if mailed in the United States

Postage will be paid by:

DATAPOINT CORPORATION
DIRECTOR OF SOFTWARE SUPPORT
8550 Datapoint Drive, Mail Station# N60
San Antonio, Texas 78284