July 21, 1959

MEMORANDUM TO: All Holders of STRETCH PROGRAMMING SYSTEM SPECIFICATIONS MANUAL

SUBJECT: Revisions

Enclosed are the latest revisions which, when inserted, will bring the STRAP-1 manual completely up-to-date. Not all pages have been revised. Each revision sheet replaces the page of the same number, with the exception of page 1-29, which is an addition.

Please note that the enclosed revisions are dated at the top of each page. All future revisions will be similarly dated. Description by date of the contents of the most up-to-date manual can then be issued from time to time to permit you to check on your receipt of all revisions.

Paul S. Herwitz, Manager
Machine Oriented Programming

PSH:HM:mc
Encl.

1.2 Expression of Machine Instructions

Machine instructions are written symbolically on the coding form described above. Normally they are entered one per line according to a prescribed format which varies with the type of instruction operation. The instructions are written with fixed mnemonic operation codes.

A Hollerith 11-0 double punch combination will be used to imply the end of a statement, so that multiple statements may be written per line. However, this character also implies the end of a comment, so that it may not be included in a comment. The 8-4 double punch (') implies the beginning of a comment. If the character ' appears in the name field, the entire card is treated as a comment and will be printed on the output listing without affecting the assembly process in any manner.

Other fields in the instruction format--addresses, modifiers, etc.--may be stated within the general symbolic forms of the system, and, when so stated, are said to constitute symbolic expressions. The order and manner in which such symbolic expressions are written down in specifying the elements of any particular instruction are dictated by a symbolic instruction format, that is, a general type which provides for the expression of a whole class of particular machine instructions.

1.2.1 Symbolic Instruction Formats

Symbolic instructions are entered in the statement field. Within this field variable length operation codes and address expressions are separated by commas and form sub-fields. A variable length modifier to either an operation or an address is enclosed in parentheses and attached to the modified sub-field. Blanks have no meaning in any field except to indicate the spacing desired on the printed output listing. The twelve symbolic instruction formats for STRAP-1 are:

FORMAT TYPE	OPERATION
1. OP(dds), A_{18}(I)	Floating point
2. OP, A_{19}(I)	Miscellaneous, unconditional branch, SIC
3. OP, J, A_{19}(I) or OP,J,A_{18}(I)	Direct index arithmetic
4. OP, J, A_{19} or $\overline{OP}$, J, A_{18}	Immediate index arithmetic
5. OP, J, $B_{19}(\overline{K})$	Count and branch
6. OP, $B_{19}(\overline{K})$	Indicator branch
7. OP(dds), A_{24}(I), OF_7(I')	VFL arithmetic, connect, convert
8. $OP_1(OP_2)$(dds), A_{24}(I), OF_7(I')	Progressive indexing
9. OP, J, A_{18}(I), A'_{18} (I)	Swap, transmit full words
10. OP, A_{24}(I), B_{19}(K)	Branch on bit
11. OP,IO(I), CW_{18}(I')	Input-output select
12. LVS, J, A, A', A'', A''', ...	Load value with sum

Definitions of the above format symbols are:

1. OP and OP_1	primary instruction operation.
2. OP_2	a secondary operation permitted only in progressive indexing and input-output.
3. A_n	an "n" -bit data address.
4. B_{19}	a 19-bit branch address.
5. I	a 4-bit index address where (0) signifies no indexing and (1.) to (15.) signifies indexing by the corresponding index register.
6. K	a 1-bit index address where no modification (0) or modification by index register 1 (1.) are the only possibilities.
7. OF_7	A 7-bit offset field.
8. dds	data description (see Section 1.2.2).
9. J	a 4-bit index address which refers to an index register as an operand. Here (0) refers to index register 0, word 16.
10. IO	input-output unit address.
11. CW_{18}	An 18-bit control word address.
12. LVS	Refers to one specific operation-Load Value with Sum.
13. primes	used to distinguish otherwise identical fields in a format.

There is a general right to left "drop-out" order for all the fields separated by commas. For example, a VFL instruction (Format 7 above) for which the offset and its index modifier are zero is written:

OP, A(I)

The comma is the major separator for the symbolic instruction types. If there are less than the maximum number of major symbolic fields in a given instruction expression (as indicated by the comma count), then the instruction is compiled as though the missing fields had been added at the end of the statement and as though they contained zeros. Such fields, whose contents are implied in a standard way by the omission of any explicit specification, are called null fields. A null field is always compiled as a zero, with the exceptions, indicated below in Section 1.2.2, of those sub-fields of a data description which express mode and byte size. Within a major field, a parenthesized sub-field may be made null by omission. Thus in the VFL example cited above, if the main index designation were to be zero but the offset and its index modifier (which in the hardware also modifies field length and byte size) were both to be one, the instruction could be written:

OP, A, 1(1.)

A major field may be null, even though other non-null fields follow it. Such is the case if nothing but the comma denoting the field termination is written. Thus in the example just shown if the offset and its modifier were both to be one but the principal address and its modifier were both to be zero, the instruction could be written:

OP, , 1(1.)

1.2.2 Data Description (dds)

The small letters "dds" enclosed in parentheses in the above formats stand for the data description field. It is established by specifying:

1. M use mode,
2. L field length, and
3. BS byte size.

These three entries appear in the above order within parentheses and are separated with commas thus, (M, L, BS). When the data description is specified in a machine instruction, it over-rules any other implied or indirectly specified data description. When it is not specified, the description is assumed to be that associated with the symbol in the principal address field of the machine instruction. If this symbol has no data description associated with it, an error condition arises.

When a string of symbols are added in an address field, the last symbol written down is the one whose data properties control those of the instruction.

A complete description of the method by which a data description may be attached to the symbol which names a piece of data is given in Section 1.3.1 under the explanation of the Data Description pseudo-operation.

The mode "M" is always specified in a data description entry. This is to say that "M" may never be a null field, so that, for example, if the first character in a data description were a comma, an error would be indicated. The seven modes are:

1. B binary
2. BU binary unsigned
3. D decimal
4. DU decimal unsigned
5. N normalized floating point
6. U unnormalized floating point
7. P Properties mode

The mnemonic "P" in the mode field of a data description has the following meaning:

(P, RIVER)

implies in an instruction that the data description associated with the symbol RIVER is to be invoked just as though it had been written out explicitly. Thus, in an instruction, the dds of RIVER would over-rule anything implied by the symbol in the major address field.

Within a data description field the usual right to left drop-out order and null field conventions hold (except, as indicated, that the mode field may not be null), so that a data description may appear in any of the following four forms:

(M)	Field length and byte size are null
(M, L)	Byte size is null
(M, , BS)	Field length is null
(M, L, BS)	

If the field length is null, a field length of 0 (effectively 64, except in the case of immediate VFL operations, where it is 24) is compiled. If the byte size is null, the compiled byte size is a function of the mode:

Mode	Standard Byte Size
D or DU	4
B	1
BU	8

1.2.3 Mnemonics

A complete list of all machine mnemonics is included in Appendix A. Both operation codes and system symbols are included in the list.

A complete list of STRAP-1 pseudo operation mnemonics is presented in Appendix B.

1.2.4 Numbers and Symbols

There are two different number systems which in general run through the STRAP-1 language, the ordinary system of real numbers and a bit-address numbering system. The ordinary real numbers are restricted in all non-data fields to be integers. Real numbers which are not integers may, of course be entered as data, but they may not take part in arithmetic expressions nor may they be symbolized, so that the general forms of the language are really limited to integers and bit addresses.

Bit Addresses consist of a pair of integers separated by a period. The integer to the left of the period specifies a word address while the integer to the right specifies a bit address. Thus, 6.32 is the decimal equivalent of either a 19 or 24-bit binary address specifying bit 32 of memory location 6- the bit preceeded by exactly 6 and one-half memory words. (Note that only the presence of a period distinguishes a bit address from an integer.)

Symbols which identify memory elements in the object program are automatically assigned bit addresses which locate these memory elements. A symbol may, however, be given the value of an integer through the use of a "synonym" pseudo-operation. Thus in general both bit addresses and integers may be symbolized. The term "integer" will be used to denote either an integral number or a symbol which takes on an integral value, and similarly so with respect to the term "bit address".

Thus, the address designation A(I) has two possible meanings:

i) If I is a bit address, then it designates an index word and is compiled in the so-called I-field.

ii) If I is an integer, then an address equal to A plus I times the field length of A is compiled.

A symbol is any sequence of eight or fewer alphabetic and numeric characters conforming to the following conditions:

1. It contains only alphanumeric characters.

2. Its first character is specifically alphabetic.

3. It appears in the name field of a program instruction by virtue of which it is "defined" and is assigned a value which is either a 24-bit binary address or an integer.

1.2.6 System Symbols

System symbols are symbols whose values have been defined by the Compiler and are therefore fixed. In all other respects, for example in relation to the conventions for legal arithmetic expressions and bit address-integer conventions, system symbols are exactly like ordinary programmer-defined symbols with the one exception that they are immediately evaluable by STRAP-1.

System symbols are identified as a special class by the prefix character "$" (which as one of the non-alphanumeric characters can never appear as part of a programmer symbol). All system symbols which stand for the addresses of special registers in memory (e.g. L, the left half of the accumulator) are bit addresses, and all others are integers or real numbers.

The appearance of the "$" character alone makes for a special system symbol which provides a standardized substitute in place of a name for the current statement. This is to say that the character "$" is a bit address which in any particular statement wherein it appears functions as though it had been defined by being written in the NAME field of that statement.

A special use of the "$" character is to prefix any operation code in this manner--$OP--. This directs the compiler to suppress any error indications which arise in connection with the compilation of this statement.

Since the actual numerical addresses which are to identify particular I/O units and channels may be chosen arbitrarily, system symbols which represent integers are provided for use in addressing I/O equipment. The numerical values of members of this set of system symbols, unlike the values of all the others, may vary from one installation to another, in order that RDR--for example--may represent the card reader channel address independently of what that address, in any particular installation, may be

I/O System Symbols are:

Symbol	Meaning
PCH	Punch (Channel Address)
PRT	Printer (Channel Address)
RDR	Reader (Channel Address)
DISK	Disk Unit (Channel Address)

Note: The arcs of a disc may be addressed by any legal symbolic integer expression, evaluated modulo 2^{12} to assure a valid arc address.

C0, C1 ...Ck	General channel addresses.
T0, T1 ....Tk	Tape Units (Unit Addresses) for a channel which includes k + 1 units.
IQS	Inquiry Station (Channel or Unit Address). This symbol may have different values depending on whether it appears in a channel address or unit address field of a symbolic select order.
CNSL	Console (Channel or Unit Address)

The system symbol mnemonics for tapes and channels are numbered in the expectation that more than one of each kind will be typical.

All of the other units named however, are also capable of plural attachment to a machine configuration, in which case numerical suffixes are added to expand the single-unit system symbol in a standard way. For example, if there are k punches for a given machine, their system symbols are: PCH0, PCH1, PCH2...PCHk - 1, where PCH0 is synonomous with PCH.

At each installation's option some system symbols--representing equipment not included in the particular system at hand--may elicit error flags on the listing.

1.2.7 Variable-in-Number Field Format

The Load Value with Sum (LVS) instruction may be written with a variable number of address fields, each of which actually picks out a single bit position within the LVS address field itself. For an LVS order, each address field may specify one of index registers 0 through 15. These fields are evaluated exactly as if they were regular index designator fields, so that index addresses may be specified in terms of either bit addresses or integers in the normal manner.

1.3 Pseudo Operations

In this section will be found itemized a number of operation codes provided for purposes of defining data and of controlling and directing the assembly process itself. Since these codes do not directly produce machine instructions in the object program, the functions which they do trigger are referred to as "pseudo operations".

The pseudo operations are grouped according to type. There are two main classes of pseudo operations:

1. Those which create memory elements.

2. Those which control the assembly process.
   a. Those which define symbols by assigning values which appear in the variable field.

   b. Those which give directions to the compiler.

The NAME field of all pseudo operations which neither create memory elements nor define symbols is ignored, with the exception of CNOP (see Section 1.3.3).

1.3.1 Pseudo Operations That Create Memory Elements

The following provide the basic means for defining and entering generalized data in the STRAP-1 language:

Mnemonic	Name	Usage
1. DD	"DATA DEFINITION"	DD (dds), $N_1, N_2, \ldots N_k$ where the bracketed "dds" is a data description prescribing the meaning of all succeeding numbers (N). The numbers N are compiled in consecutive fields and any symbol appearing in the NAME field of the DD statement applies to the first such field.

The data description (dds) is identical in form and content to that described in Section 1.2.2, that is, to the data description which may be used when writing an individual instruction except the P mode is not omitted in this or any other pseudo operation. Thus a description may be given with a number at the point of definition of the number itself, or may be given at the point of reference as part of an instruction referring to the number. The relation between these two different points of possible definition is as follows:

When the data description is given by a DD statement (or other data defining operation), the description is invoked whenever the symbol appearing in the NAME field of the DD statement is used in the principal address field of an instruction. The instruction mode, and--in the case of a VFL order--the field length and byte size are supplied by this data description which is logically affixed to the name of the DD statement.

Such a description set down at the point of symbol definition is over-ruled by a description appearing in an instruction referring to the symbol. Whenever an over-ruling description appears in the data description field of an instruction, the entire description which was given at the point of definition of the address symbol is over-ruled. Thus the statement:

```
OP (BU), JOE
```

causes the binary and unsigned modifiers to be compiled along with an

implicitly defined field length of 64 and a byte size of 8, regardless of the description occurring in the statement in which JOE appeared in the NAME field. Over-ruling is strictly local and applies only to the instruction at hand.

If symbols are used in defining either the field length or byte size sub-fields of a DD statement's data description, the symbols must be fully defined when the compiler encounters the DD statement on the second pass. This requirement is not imposed on the data description of an instruction since, in that instance, no assignment of memory space is dependent on the contents of the sub-fields.

Symbols which name instructions themselves are automatically imbued with data descriptions. Specifically, instruction-naming symbols are given field lengths equal to the lengths of the particular instructions named (i.e. either 32 or 64), and are defined as unsigned binary with byte size 8.

System symbols whose values are the bit addresses of special registers in memory also have data descriptions which have been fixed by the compiler (although, as with ordinary symbols, these descriptions may be over-ruled by the data description fields of instructions). Specifically, system symbols representing memory registers are binary unsigned, have field lengths equal to the lengths of their represented registers, and have byte size 8.

2. XW "INDEX WORD" XW, VALUE, COUNT, REFILL, FLAG

The location counter is rounded to the next full word. The contents of the four symbolic fields following the operation are converted and compiled in an index word format. FLAG denotes the machine field comprised of bits 25, 26 and 27. An expression in the FLAG field of an XW statement is therefore evaluated modulo 2^3.

> Note: Bit 24 of the word format is taken to be the VALUE sign position. A negative sign is interpreted in two's complement form in the usual way for all other fields.

3. VF "VALUE FIELD" VF, VALUE

The location counter is rounded to the next half word. The contents of VALUE are compiled as a 24-bit plus sign quantity in positions 0-24 of the next half word. The location counter stands at bit 25 at the end of the operation.

4. CF "COUNT FIELD" CF, COUNT

The location counter is rounded to the next half word. The contents of the COUNT field are compiled as an 18-bit integer in positions 0-17. The location counter stands at bit 18 at the end of the operation.

5. RF "REFILL FIELD" RF, REFILL

This pseudo operation is the same as CF.

NOTE: The last four operations (the index word pseudo operations) defined above are given data descriptions by the compiler, just as though they had been defined by DD statements. Specifically, the index elements created by these orders have had the following data descriptions affixed automatically, and cannot be over-ruled in the pseudo-op statement:

OPERATION	DATA DESCRIPTION
XW	(BU)
VF	(B,25)
CF or RF	(BU,18)

6. CW "CONTROL WORD" CW(OP),ADDRESS,COUNT, CHAIN ADDRESS

The pseudo operation CW employs a special symbolic format as illustrated above and defined initially in Section 1.2.1. A set of secondary operations is provided--whose members are expressed as parenthesized secondary operations in the manner of "(OP)" above--with the purpose of providing mnemonics for control word functions:

	Multiple Bit	Chain Bit
CR "COUNT WITHIN RECORD"	0	0
CCR "CHAIN COUNTS WITH-IN RECORD"	0	1
CD "COUNT DISREGARDING RECORD"	1	0
CDSC"COUNT DISREGARDING RECORD,SKIP AND CHAIN"	1	1

1.3.1.1 The Form of N in a Data Definition Statement

All data falls under the category of one of the six modes of the data description field: N, U, B, BU, D, and DU. The numbers $N_1 \ldots N_K$ are expressed in the form:

$$\pm \text{XXX.XX}$$

and may optionally have other quantities following them which are identified and separated from the main number by declension characters:

E ± i — The integer "i" is taken as a decimal exponent of the preceding number. Over-lapping facilities for specifying an exponent "Ei" are provided in the sense that the decimal point in the number itself also indicates a decimal exponent. If no point occurs explicitly, the number is taken to be an integer.

Si — The positive octal integer "i" is compiled as the sign byte of the preceding number. If either the sign of the main number or i implies a negative sign bit, the sign byte sign position is made negative.

X ± i — The decimal integer "i" is compiled as a machine exponent of a floating point number. It over-rules and replaces the computed exponent, which is completely eradicated by the replacement process.

NOTE: The data entries in a DD statement are restricted to real numbers only. Bit addresses are not permitted. Integers are of course allowed as a special case of real numbers, but they may not be symbolized.

Floating point data is always compiled in addressable full words; the location counter is rounded up, if necessary, to the next full word address in order to meet this end. This is an instance of a general STRAP 1 principal: a machine format which ordinarily depends in use on the fact that the 24-bit address of the lead bit ends in a string of zeroes of some definite length causes the compiler to round the location counter appropriately. Thus:

1) Instructions always start at either half or full word bit addresses.

1.3.1.2 The Entry Mode

The data description field represents a kind of generalized use mode for the data, in that properties specified in this field are translated into bits and numbers which are compiled into machine instructions referring to the data. A corresponding field called the entry mode is available to specify properties which describe the source language information and its form, but which properties are not themselves compiled into the object program.

The entry mode may be employed in one of two ways:

a) An entry mode may be used to specify the properties of any symbolic field (except the "field" occupied solely by the operation mnemonics) by being placed, enclosed in parentheses, as the first item in the field.

b) An entry mode may also be used to specify the properties of all the data in a DD or DDI statement. When used in this fashion, it is enclosed in parentheses and appears before the DD or DDI op code in the operation field. The mode is more general in form in its usage in connection with the data of a DD or DDI statement, as it may in this instance--but only in this instance--designate that alphabetic information is to be compiled:

ENTRY MODE	MEANING
(AX)	"A" signifies that the following information is 704-9 alphabetic (BCD as it appears on tape), and the letter X is a special end-of-statement mark for this statement only. The end of statement character is not itself compiled. The special end-character may not be:) ' (8-4) 11-0 blank
(IQSX)	The code IQS implies the IQS alphabetic code, and this entry mode designation is otherwise the same as the preceding. When IQS is specified in an entry mode, only those IQS characters which also exist in Hollerith may be entered.

(Fi) In DD and DDI binary-mode statements, the number of binary fractional bits is specified in the entry mode by means of the letter F followed by a decimal integer i which is the number of fractional bits.

(F6) XX.XXX

Entry modes may not appear in a manner that would cause parentheses within parentheses. An entry mode may appear as the first element of any field in the DD or DDI statement, in which case it functions as a normal field entry mode. When contradictory properties (for instance, two differing radixes) are implied by the statement and field entry modes, the field mode over-rules for the case of the particular field on hand.

NOTE: Both the statement entry mode and the field entry modes in a DD or DDI statement apply only to the pure number part of the data. All other quantities which may be joined to the data by special declensions (e.g. S for sign byte) are regarded as separate fields with respect to the entry mode, and these fields will have no provision for a separate entry mode in STRAP-1. Moreover, if the entry mode indicates a radix different from 10, only integers may be entered as data.

There are two kinds of designators which may appear in any entry mode expression:

a) Any of the digits 2 through 10 may be used to indicate a radix. All numerical quantities governed by the entry mode--whether real numbers, integers, or bit addresses--are then interpreted in the specified radix. The source language radix is 10 throughout the system unless otherwise specified (and after the declension character S where the base 8 is used).

b) An integer preceded by a point not exceeding 63 (or 31 in a half word instruction) has the following meaning in the entry mode: that the field following the entry mode is parenthetical in nature and is to be evaluated and compiled with the specified bit address serving as the bit address of the rightmost position of the field. The field is added by a logical OR so that it may be combined with other fields of the statement or other parenthetical OR fields. The first bit of the statement is counted as bit 0. Although the parenthetical field may cross field-lines within a statement, it may not cross statement-lines. That is, if the bit address is specified as ".n", the parenthetical expression has a field length of $n + 1$ and is evaluated modulo $2^n + 1$.

All parenthetical fields are regarded as unsigned, so that a negative number is compiled as the complement, re $2^n + 1$, of the magnitude of the number.

The field following an entry mode containing a bit address is terminated either by the end-of-field character of the statement field in which the parenthetical OR field falls (i.e., within the source language--the parenthetical field may cross field lines within the object language but by its very nature is always specified within the bounds of some other field in the source language) or by the beginning-field character for some other field.

Radix designators are permitted in parenthetical OR fields, are separated by commas from the bit designation, and the two may be in any order:(.32, 8) signifies an octal field to be terminated at bit 32.

Parenthetical expressions are permitted within a DD statement, and the bit address is measured from the previous comma forward. Parenthetical expressions may contain anything that goes in a normal address field (except periods), but may not have other information--like real numbers or alphabetic characters--which are permitted in a DD or DDI statement. Parenthetical expressions are not permitted in any statement which does not compile memory space, nor in a DR statement.

The parenthetical field ignores both the field structure and any data description associated with the statement in which it appears. Similarily, any data description associated with a symbol appearing in a parenthetical field has no effect in this usage of the symbol. All numbers--including real numbers--which appear in a parenthetical field are converted to an internal binary format, never to decimal or floating point.

1.3.2 Pseudo Operations Which Define Symbols

It can be said that almost all pseudo operations (excluding SLC, CNOP, etc.) define symbols in the standard manner -- any symbol appearing in the name field will be assigned the current value of the location counter. Grouped under the present category of pseudo operations are those which define symbols in other than the usual manner.

1. DDI "DATA DEFINITION IMMEDIATE"

This pseudo operation is identical to DD except with respect to the following points:

(1) Like SYN (see below), it is purely definitive in character.
(2) Only one major field follows the operation field of the statement.
(3) If no field length is specified, a field length of 24 is implied.
(4) If the length of a string of alphabetic characters exceeds the field length, the excessive low-order characters are lost and an error indication is given.
(5) The compiled field -- less than or equal to 24 bits in length--is inserted within a 24-bit field within the symbol table and left justified.

2. SYN "SYNONYM" A SYN, Y

The operation "Synonym" (SYN) may define a new symbol in terms of a symbolic expression representing either a bit address or an integer, with the restriction--as with SLC -- that the expression be fully defined although this may possibly mean as late as pass 3 of STRAP-1. When one writes:

```
A  SYN, Y
```

the meaning of the newly defined symbol "A" is that whenever A is written in the program the effect is the same as if Y had been written. The meaning of SYN is always one of exact substitution. Thus data properties associated with Y and its bit address-or-integer classification are transferred to A. SYN statements are permitted to have their own data description field, as well.

1.3.3 Pseudo Operations Which Give Directions To The Compiler

Mnemonic	Name	Usage
1) SLC	"SET LOCATION COUNTER"	A SLC, Y

This operation resets the location counter to the value (bit address) taken on by Y, where Y is any legal symbolic expression. The name A applies to the subsequently defined memory element whose first bit is located at Y. Y must be evaluable by the time it is encountered in pass 2 of STRAP-1. Although Y may be an absolute number, its absolute meaning may not be preserved from STRAP-1 to STRAP-2. In STRAP-1 an absolute origin will be positioned relative to a program area beginning with machine location 0. In STRAP-2 the beginning of the program area will normally be supplied independently of the assembly deck and may differ from 0.

The pseudo operation "Set Location Counter" must contain a bit address expression whose value is positive. An integer which appears in the variable field of an SLC instruction is added in as in a 24-bit address field, i.e. as an integral number of bits, and an error warning is given.

2) END	"END"	A END, Y

A card with the operation code END signals the end of an assembly and must be included as the last card of each symbolic program deck. If the END card is missing, one is supplied by STRAP-1. A branch card is then punched with the output deck with an address Y, so that the instruction located at Y will be the first program order executed.

The END statement also functions as an origin-setting statement for the memory assignments given to all symbols which are undefined. A symbol is undefined if it appears somewhere in the program but never appears in the NAME field of any statement. All occurrences of such a symbol are flagged as possible errors. The symbol is assigned a full memory word in the block whose origin is equal to the value of the location counter when the END statement is encountered (possibly rounded up to obtain an integral full word address) and the symbol is given a normalized floating point data description.

3) CNOP "CONDITIONAL NO OPERATION"

The pseudo operation CNOP is used to insure that the instruction immediately following the CNOP will be assigned a full word address by the compiler.

CNOP examines the location counter. If the counter is already set to a full word address, the compiler ignores the CNOP. If, however, the instruction counter is set to a half word address, the CNOP instruction directs the compiler to advance the counter 32 bits (one half word) to the next full word address. This is accomplished by compiling the machine instruction NOP, which is a half word instruction. Any symbol appearing in the name field is assigned a full word address when the CNOP is ignored, or a half word address when a NOP is compiled.

4) TLB "TERMINATE LOADING AND BRANCH"

The pseudo operation "Terminate Loading and Branch" is similar to an "END" statement with one major distinction--TLB does not stop the assembly process. Therefore, TLB may be used at any point in a symbolic deck where a branch card is desired. The branch card thus produced will interrupt the loader when encountered in a binary deck and transfer control to the instruction at location Y.

5) EXT "EXTRACT" A EXT (I,J) STATEMENT

The "Extract" pseudo operation has the following meaning:
I and J are integers or integer expressions. STATEMENT is assembled by the processor as though it were to be compiled. The field beginning at bit I and ending at bit J within the assembled statement is then extracted, and compiled. Any symbol A in the NAME field of the EXT order is attached to this compiled quantity. A data description is attached to the symbol as though it had been written:

(BU, J - I + 1, 8)

I and J can also be bit addresses.

If EXT is used to specify the extraction of anything beyond the range of the single statement which follows it, zeroes are added up to 64.

6) DR "DATA RESERVATION" A DR (dds), (N)

The DR operation causes N fields of the kind described in the data description--(dds)--field to be reserved, i.e. the instruction location counter is skipped forward a quantity in bits equal to the product of N and the field length specified in (dds). The symbol A if any, appearing in the NAME field of the DR statement is attached to the first such field. The description specified in (dds) is in turn attached to the symbol and is invoked--in the same fashion as with a DD or DDI statement--whenever the symbol appears as a principal address.

An array is specified in the form:

NAME DR (dds), (I, J, K)

where I, J and K must be integers in symbolic or numeric form. In fact, an array parameter may be specified by any integer-valued expression.

Although a fifteen dimensional array is the largest which can be specified in STRAP-1, arrays involving fewer parameters can also be described.

Thus, as seen from the discussion in Section 1.2.4, to apply index word I to the second element of a one dimensional array A, one writes:

A (1) (I)

where I must be a bit address.

7) PRNS "PRINT SINGLE-SPACED" PRNS

This pseudo operation causes the assembly listing to be printed with single spacing. Double spacing is the normal mode unless PRNS is written.

8) PRND "PRINT DOUBLE-SPACED" PRND

This pseudo-op restores printing to the normal double spacing condition after the use of a PRNS.

9) PUNFUL "PUNCH FULL CARDS" PUNFUL

Full cards (80 columns) are punched, without checksum, FWA, ID, etc.

10) PUNNOR "PUNCH NORMALLY" PUNNOR

This pseudo-op restores normal punching after the use of a PUNFUL.

11) SKIP "SKIP PAPER" SKIP,i

If $i = 0$, the assembly listing will restore the paper immediately. If $i \neq 0$, one half page will be skipped.

12) PUNID "PUNCH ID" PUNID,XXXXXXXX

The first eight characters following the comma are punched in columns 73-80 of the binary deck produced by the assembly program. This card is used to identify the assembly.

13) PRNID "PRINT ID" PRNID, COMMENT

When PRNID is encountered anything written after the comma is immediately printed on line and on the output tape as well. PRNID provides a means of heading the assembly listing with such information as problem name. programmer, etc.

ALPHABETIC LIST OF SYSTEM SYMBOLS

				WORD NO.	BIT ADDRESS
$	AD	2	ADDRESS INVALID	11	16
$	AE	2	ACCUMULATOR EQUAL	11	61
$	AH	2	ACCUMULATOR HIGH	11	62
$	AL	2	ACCUMULATOR LOW	11	60
$	AOC	1	ALL ONES COUNT	7	44-50
$	BC	1	BOUNDARY CONTROL	3	57
$	BTR	2	BINARY TRANSIT	11	39
$	CA	1	CHANNEL ADDRESS	5	12-18
$	CBJ	2	CHANNEL BUSY REJECT	11	8
$	CNSL	1	CONSOLE		
$	CPUS	1	CPU SIGNAL	11	5
$	CPU	2	OTHER CPU	6	0-18
$	CS	2	CHANNEL SIGNAL	11	13
$	CX	1	CHANNEL X (X IS A NUMERICAL DESIGNATION)		
$	DF	2	DATA FETCH	11	20
$	DISK	1	DISK		
$	DS	2	DATA STORE	11	19
$	DTR	2	DECIMAL TRANSIT	11	40
$	E	12	e		
$	EE	2	END EXCEPTION	11	11
$	EK	2	EXCHANGE CONTROL CHECK	11	3
$	EKJ	2	EXCHANGE CHECK REJECT	11	6
$	EOP	2	END OF OPERATION	11	12
$	EPGK	2	EXCHANGE PROGRAM CHECK	11	9
$	EXE	2	EXECUTE EXCEPTION	11	18
$	FT	1	FACTOR	14	0-63
$	IA	1	INTERRUPTION ADDRESS	2	0-17
$	IF	2	INSTRUCTION FETCH	11	21
$	IK	2	INSTRUCTION CHECK	11	1
$	IJ	2	INSTRUCTION REJECT	11	2
$	IND	1	INDICATORS	11	0-63
$	IQS	1	INQUIRY STATION		
$	IR	2	IMAGINARY ROOT	11	25
$	IT	1	INTERVAL TIMER	1	0-18
$	L	1	LEFT HALF OF ACCUMULATOR	8	0-63
$	LB	1	LOWER BOUNDARY	3	32-49
$	LC	2	LOST CARRY	11	22
$	LS	2	LOST SIGNIFICANCE	11	26
$	LZC	1	LEFT ZEROS COUNT	7	17-23
$	M	12	$\log_{10} e$		

ALPHABETIC LIST OF SYSTEM SYMBOLS

				WORD NO.	BIT ADDRESS
$	MASK	1	MASK	12	21-49
$	MB	1	MAINTENANCE BITS	4	0-63
$	MK	2	MACHINE CHECK	11	0
$	MOP	2	TO-MEMORY OPERATION	11	55
$	N	12	$\log_e 2$		
$	NM	2	NOISY MODE	11	63
$	OP	2	OPERATION INVALID	11	15
$	PCH	1	PUNCH		
$	PF	2	PARTIAL FIELD	11	23
$	PGO..PG6	2	PROGRAM INDICATORS	11	41-47
$	PI	12	π		
$	PRT	1	PRINTER		
$	PSH	2	PREPARATORY SHIFT GREATER THAN 48	11	27
$	R	1	RIGHT HALF OF ACCUMULATOR	9	0-63
$	RDR	1	READER		
$	RGZ	2	RESULT GREATER THAN ZERO	11	58
$	RLZ	2	RESULT LESS THAN ZERO	11	56
$	RM	1	REMAINDER	13	0-63
$	RN	2	RESULT NEGATIVE	11	59
$	RU	2	REMAINDER UNDERFLOW	11	34
$	RZ	2	RESULT ZERO	11	57
$	SB	1	SIGN BYTE	10	0-7
$	TC	1	TIME CLOCK	1	28-63
$	TF	2	T FLAG	11	35
$	TR	1	TRANSIT	15	0-63
$	TS	2	TIME SIGNAL	11	4
$	TX	1	TAPE X (X IS A NUMERICAL DESIGNATION)		
$	UB	1	UPPER BOUNDARY	3	0-17
$	UF	2	U FLAG	11	36
$	UK	2	UNIT CHECK	11	10
$	UNRJ	2	UNIT NOT READY REJECT	11	7
$	USA	2	UNENDED SEQUENCE OF ADDRESSES	11	17
$	VF	2	V FLAG	11	37
$	XO	1	INDEX ZERO	16	0-63
$	X1	1	INDEX ONE	17	0-63
$	X2	1	INDEX TWO	18	0-63
$	X3	1	INDEX THREE	19	0-63
$	X4	1	INDEX FOUR	20	0-63
$	X5	1	INDEX FIVE	21	0-63
$	X6	1	INDEX SIX	22	0-63

I. OVER-ALL LIST OF MNEMONICS

B. OPERATIONS

V	+	3	ADD
F	+	6	ADD
V	+MG	3	ADD TO MAGNITUDE
F	+MG	6	ADD TO MAGNITUDE
V	-	3	SUBTRACT
F	-	6	SUBTRACT
V	-MG	3	SUBTRACT FROM MAGNITUDE
F	-MG	6	SUBTRACT FROM MAGNITUDE
V	*	4	MULTIPLY
F	*	7	MULTIPLY
V	* +		MULTIPLY AND ADD
F	* +		MULTIPLY AND ADD
F	*A +		MULTIPLY ABSOLUTE AND ADD
V	*I +		MULTIPLY IMMEDIATE AND ADD
V	*N +		MULTIPLY NEGATIVE AND ADD
F	*N +		MULTIPLY NEGATIVE AND ADD
F	*NA +		MULTIPLY NEGATIVE ABSOLUTE AND ADD
V	*NI +		MULTIPLY NEGATIVE IMMEDIATE AND ADD
V	/	4	DIVIDE
F	/	7	DIVIDE
M	B		BRANCH
B	BB		BRANCH ON BIT
B	BB1		BRANCH ON BIT AND SET TO ONE
B	BBN		BRANCH ON BIT AND NEGATE
B	BBZ		BRANCH ON BIT AND ZERO
M	BD		BRANCH DISABLED
M	BE		BRANCH ENABLED
M	BEW		BRANCH ENABLED AND WAIT
M	BR		BRANCH RELATIVE
E	BS		BACKSPACE
E	BSFL		BACKSPACE FILE
B	BZB		BRANCH ON ZERO BIT
B	BZB1		BRANCH ON ZERO BIT AND SET TO ONE
B	BZBN		BRANCH ON ZERO BIT AND NEGATE
B	BZBZ		BRANCH ON ZERO BIT AND ZERO
V	C	10	CONNECT
I	C + I		ADD IMMEDIATE TO COUNT
I	C - I		SUBTRACT IMMEDIATE FROM COUNT
C	CB	8	COUNT AND BRANCH
C	CBR	8	COUNT, BRANCH AND REFILL
C	CBZ	8	COUNT AND BRANCH ON ZERO COUNT
C	CBZR	8	COUNT, BRANCH ON ZERO COUNT AND REFILL
E	CCW		COPY CONTROL WORD
V	CM	10	CONNECT TO MEMORY

V	CT	10	CONNECT FOR TEST
E	CTL		CONTROL
V	CV	5	CONVERT
F	D +	6	ADD DOUBLE
F	D + MG	6	ADD DOUBLE TO MAGNITUDE
F	D -	6	SUBTRACT DOUBLE
F	D - MG	6	SUBTRACT DOUBLE FROM MAGNITUDE
V	DCV	5	CONVERT DOUBLE
F	DL	7	LOAD DOUBLE
F	DLWF	7	LOAD DOUBLE WITH FLAG
F	D*	7	MULTIPLY DOUBLE
F	D/	7	DIVIDE DOUBLE
F	E +	6	ADD TO EXPONENT
F	E + AI		ADD ABSOLUTE IMMEDIATE TO EXPONENT
F	E + I		ADD IMMEDIATE TO EXPONENT
F	E -	6	SUBTRACT FROM EXPONENT
F	E - AI		SUBTRACT ABSOLUTE IMMEDIATE FROM EXPONENT
F	E - I		SUBTRACT IMMEDIATE FROM EXPONENT
E	ERG		ERASE GAP
E	EVEN		EVEN PARITY
M	EX		EXECUTE
M	EXIC		EXECUTE INDIRECT AND COUNT
F	F +	6	ADD TO FRACTION
F	F -	6	SUBTRACT FROM FRACTION
E	GONG		GONG
E	HD		HIGH DENSITY
V	K	4	COMPARE
F	K	7	COMPARE
I	KC		COMPARE COUNT
I	KCI		COMPARE COUNT IMMEDIATE
V	KE	4	COMPARE IF EQUAL
V	KF	4	COMPARE FIELD
V	KFE	4	COMPARE FIELD IF EQUAL
V	KFR	4	COMPARE FIELD FOR RANGE
E	KLN		CHECK LIGHT ON
F	KMG	7	COMPARE MAGNITUDE
F	KMGR	7	COMPARE FIELD FOR RANGE
V	KR	4	COMPARE FOR RANGE
F	KR	7	COMPARE FOR RANGE
I	KV		COMPARE VALUE
I	KVI		COMPARE VALUE IMMEDIATE
I	KVNI		COMPARE VALUE NEGATIVE IMMEDIATE
V	L	4	LOAD

F	L	7	LOAD
I	LC		LOAD COUNT
I	LCI		LOAD COUNT IMMEDIATE
V	LCV	4	LOAD CONVERTED
E	LD		LOW DENSITY
V	LF		LOAD FIELD
V	LFT	4	LOAD FACTOR
F	LFT	7	LOAD FACTOR
E	LOC		LOCATE (SAME AS SELECT UNIT)
I	LR		LOAD REFILL
I	LRI		LOAD REFILL IMMEDIATE
I	LV		LOAD VALUE
I	LVE		LOAD VALUE EFFECTIVE
I	LVI		LOAD VALUE IMMEDIATE
I	LVNI		LOAD VALUE NEGATIVE IMMEDIATE
I	LVS		LOAD VALUE WITH SUM
I	LX		LOAD INDEX
V	LTRCV	4	LOAD TRANSIT CONVERTED
V	LTRS	4	LOAD TRANSIT AND SET
V	LWF	4	LOAD WITH FLAG
F	LWF	7	LOAD WITH FLAG
V	M+	3	ADD TO MEMORY
F	M+	6	ADD TO MEMORY
V	M+1		ADD ONE TO MEMORY
F	M+A		ADD TO ABSOLUTE MEMORY
V	M+MG	3	ADD MAGNITUDE TO MEMORY
F	M+MG	6	ADD MAGNITUDE TO MEMORY
V	M-		SUBTRACT FROM MEMORY
F	M-		SUBTRACT FROM MEMORY
V	M-1		SUBTRACT ONE FROM MEMORY
F	M-A		SUBTRACT FROM ABSOLUTE MEMORY
V	M-MG	3	SUBTRACT MAGNITUDE FROM MEMORY
F	M-MG	6	SUBTRACT MAGNITUDE FROM MEMORY
M	NOP		NO OPERATION
E	ODD		ODD PARITY
M	R		REFILL
M	RCZ		REFILL ON COUNT ZERO
E	RD		READ
E	REL		RELEASE
E	REW		REWIND
E	RLF		RESERVED LIGHT OFF
E	RLN		RESERVED LIGHT ON
I	RNX		RENAME
F	R/	7	RECIPROCAL DIVIDE
I	SC		STORE COUNT

E	SEOP	11	SUPPRESS END OF OPERATION
V	SF		STORE FIELD
F	SHF	7	SHIFT FRACTION
F	SHFL		SHIFT FRACTION LEFT (SAME AS SHFA)
F	SHFR		SHIFT FRACTION RIGHT (SAME AS SHFNA)
M	SIC		STORE INSTRUCTION COUNTER IF
F	SLO	7	STORE LOW ORDER
E	SP		SPACE
E	SPFL		SPACE FILE
F	SNRT	6	STORE NEGATIVE ROOT
I	SR		STORE REFILL
V	SRD	5	STORE ROUNDED
F	SRD	7	STORE ROUNDED
F	SRT	6	STORE ROOT
V	ST	5	STORE
F	ST	7	STORE
E	SU		SELECT UNIT (SAME AS LOCATE)
I	SV		STORE VALUE
I	SVA		STORE VALUE IN ADDRESS
T	SWAP		SWAP
T	SWAPI		SWAP IMMEDIATE
T	SWAPB		SWAP BACKWARD
T	SWAPBI		SWAP BACKWARD IMMEDIATE
I	SX		STORE INDEX
T	T		TRANSMIT
T	TI		TRANSMIT IMMEDIATE
T	TB		TRANSMIT BACKWARD
T	TBI		TRANSMIT BACKWARD IMMEDIATE
I	V +		ADD TO VALUE
I	V + I	9	ADD IMMEDIATE TO VALUE
I	V + C		ADD TO VALUE AND COUNT
I	V + CR		ADD TO VALUE, COUNT AND REFILL
I	V + IC	9	ADD IMMEDIATE TO VALUE AND COUNT
I	V + ICR	9	ADD IMMEDIATE TO VALUE, COUNT AND REFILL
I	V - I	9	SUBTRACT IMMEDIATE FROM VALUE
I	V - IC	9	SUBTRACT IMMEDIATE FROM VALUE AND COUNT
I	V - ICR	9	SUBTRACT IMMEDIATE FROM VALUE, COUNT AND REFILL
E	W		WRITE
E	WEF		WRITE END-OF-FILE
M	Z		STORE ZERO

II. LIST OF MNEMONICS BY TYPE

A. FLOATING POINT

F	+	6	ADD
F	+MG	6	ADD TO MAGNITUDE
F	-	6	SUBTRACT
F	-MG	6	SUBTRACT FROM MAGNITUDE
F	*	7	MULTIPLY
F	* +		MULTIPLY AND ADD
F	*A +		MULTIPLY ABSOLUTE AND ADD
F	*N +		MULTIPLY NEGATIVE AND ADD
F	*NA +		MULTIPLY NEGATIVE ABSOLUTE AND ADD
F	/	7	DIVIDE
M	B		BRANCH
M	BD		BRANCH DISABLED
M	BE		BRANCH ENABLED
M	BEW		BRANCH ENABLED AND WAIT
M	BR		BRANCH RELATIVE
F	D +	6	ADD DOUBLE
F	D +MG	6	ADD DOUBLE TO MAGNITUDE
F	D -	6	SUBTRACT DOUBLE
F	D -MG	6	SUBTRACT DOUBLE FROM MAGNITUDE
F	DL	7	LOAD DOUBLE
F	DLWF	7	LOAD DOUBLE WITH FLAG
F	D*	7	MULTIPLY DOUBLE
F	D/	7	DIVIDE DOUBLE
F	E +	6	ADD TO EXPONENT
F	E +AI		ADD ABSOLUTE IMMEDIATE TO EXPONENT
F	E +I		ADD IMMEDIATE TO EXPONENT
F	E -	6	SUBTRACT FROM EXPONENT
F	E -AI		SUBTRACT ABSOLUTE IMMEDIATE FROM EXPONENT
F	E -I		SUBTRACT IMMEDIATE FROM EXPONENT
M	EX		EXECUTE
M	EXIC		EXECUTE INDIRECT AND COUNT
F	F +	6	ADD TO FRACTION
F	F -	6	SUBTRACT FROM FRACTION
F	K	7	COMPARE
F	KMG	7	COMPARE MAGNITUDE
F	KMGR	7	COMPARE MAGNITUDE FOR RANGE
F	KR	7	COMPARE FOR RANGE
F	L	7	LOAD
F	LFT	7	LOAD FACTOR
F	LWF	7	LOAD WITH FLAG

F	M+	6	ADD TO MEMORY
F	M+A		ADD TO ABSOLUTE MEMORY
F	M+MG	6	ADD MAGNITUDE TO MEMORY
F	M-		SUBTRACT FROM MEMORY
F	M-A		SUBTRACT FROM ABSOLUTE MEMORY
F	M-MG	6	SUBTRACT MAGNITUDE FROM MEMORY
M	NOP		NO OPERATION
M	R		REFILL
M	RCZ		REFILL ON COUNT ZERO
F	R/	7	RECIPROCAL DIVIDE
F	SHF	7	SHIFT FRACTION
F	SHFL		SHIFT FRACTION LEFT (SAME AS SHFA)
F	SHFR		SHIFT FRACTION RIGHT (SAME AS SHFNA)
M	SIC		STORE INSTRUCTION COUNTER IF
F	SLO	7	STORE LOW ORDER
F	SNRT	6	STORE NEGATIVE ROOT
F	SRD	7	STORE ROUNDED
F	SRT	6	STORE ROOT
F	ST	7	STORE
M	Z		STORE ZERO

II LIST OF MNEMONICS BY TYPE

B. I/O SELECTS

E	BS		BACKSPACE
E	BSFL		BACKSPACE FILE
E	CCW		COPY CONTROL WORD
E	CTL		CONTROL
E	ERG		ERASE GAP
E	EVEN		EVEN PARITY
E	GONG		GONG
E	HD		HIGH DENSITY
E	KLN		CHECK LIGHT ON
E	LD		LOW DENSITY
E	LOC		LOCATE (SAME AS SELECT UNIT)
E	ODD		ODD PARITY
E	RD		READ
E	REL		RELEASE CHANNEL
E	REW		REWIND
E	RLF		RESERVED LIGHT OFF
E	RLN		RESERVED LIGHT ON
E	SEOP	11	SUPPRESS END OF OPERATION
E	SP		SPACE
E	SPFL		SPACE FILE
E	SU		SELECT UNIT (SAME AS LOCATE)
E	W		WRITE
E	WEF		WRITE END OF FILE

II LIST OF MNEMONICS BY TYPE

C. VFL

V	+	3	ADD
V	+ MG	3	ADD TO MAGNITUDE
V	-	3	SUBTRACT
V	- MG	3	SUBTRACT FROM MAGNITUDE
V	*	4	MULTIPLY
V	* +		MULTIPLY AND ADD
V	*I +		MULTIPLY IMMEDIATE AND ADD
V	*N +		MULTIPLY NEGATIVE AND ADD
V	*NI +		MULTIPLY NEGATIVE IMMEDIATE AND ADD
V	/	4	DIVIDE
V	C	10	CONNECT
V	CM	10	CONNECT TO MEMORY
V	CT	10	CONNECT FOR TEST
V	CV	5	CONVERT
V	DCV	5	CONVERT DOUBLE
V	K	4	COMPARE
V	KE	4	COMPARE IF EQUAL
V	KF	4	COMPARE FIELD
V	KFE	4	COMPARE FIELD IF EQUAL
V	KFR	4	COMPARE FIELD FOR RANGE
V	KR	4	COMPARE FOR RANGE
V	L	4	LOAD
V	LCV	4	LOAD CONVERTED
V	LF		LOAD FIELD
V	LFT	4	LOAD FACTOR
V	LTRCV	4	LOAD TRANSIT CONVERTED
V	LTRS	4	LOAD TRANSIT AND SET
V	LWF	4	LOAD WITH FLAG
V	M +	3	ADD TO MEMORY
V	M + 1		ADD ONE TO MEMORY
V	M+MG	3	ADD MAGNITUDE TO MEMORY
V	M -		SUBTRACT FROM MEMORY
V	M -1		SUBTRACT ONE FROM MEMORY
V	M -MG	3	SUBTRACT MANGITUDE FROM MEMORY
V	SF		STORE FIELD
V	SRD	5	STORE ROUNDED
V	ST	5	STORE

H. SYSTEM SYMBOLS THAT ARE BIT ADDRESSES

LOACTION WORD NO.	BIT ADDRESS		SYSTEM SYMBOL MNEMONIC	NAME
			INSTRUCTION EXCEPTION	
11	15	$	OP	OPERATION INVALID
11	16	$	AD	ADDRESS INVALID
11	17	$	USA	UNENDED SEQUENCE OF ADDRESSES
11	18	$	EXE	EXECUTE EXCEPTION
11	19	$	DS	DATA STORE
11	20	$	DF	DATA FETCH
11	21	$	IF	INSTRUCTION FETCH
			RESULT EXCEPTION	
11	22	$	LC	LOST CARRY
11	23	$	PF	PARTIAL FIELD
11	24	$	ZD	ZERO DIVISOR
			RESULT EXCEPTION-FLOATING POINT	
11	25	$	IR	IMAGINARY ROOT
11	26	$	LS	LOST SIGNIFICANCE
11	27	$	PSH	PREPARATORY SHIFT GREATER THAN 48
11	28	$	XPO	EXPONENT OVERFLOW ($EXP \geqslant 2^{11}$)
11	29	$	XPH	EXPONENT HIGH ($2^{10} \leqslant EXP < 2^{11}$)
11	30	$	XPM	EXPONENT MEDIUM ($2^{8} \leqslant EXP < 2^{10}$)
11	31	$	XPL	EXPONENT LOW ($2^{5} \leqslant EXP < 2^{8}$)
11	32	$	XPN	EXPONENT HIGH NEGATIVE ($-2^{11} < EXP \leqslant 2^{10}$)
11	33	$	XPU	EXPONENT UNDERFLOW ($EXP \leqslant -2^{11}$)
11	34	$	RU	REMAINDER UNDERFLOW
			FLAGGING	
11	35	$	TF	T FLAG
11	36	$	UF	U FLAG
11	37	$	VF	V FLAG
11	38	$	XF	INDEX FLAG
			TRANSIT OPERATIONS	
11	39	$	BTR	BINARY TRANSIT
11	40	$	DTR	DECIMAL TRANSIT
11	41-47	$	PGO...PG6	PROGRAM INDICATORS

7. This floating point operation code may have the following suffixes:

   N Negative

   A Absolute

   NA Negative Absolute

8. Count and Branch operation may have the following suffixes:

   + Add one to value

   - Subtract one from value

   H Add half to value

9. This operation code may be used to indicate either an immediate indexing operation or the secondary operation of any VFL instruction.

10. This operation mnemonic specifies potentially 16 connect orders. Four binary digits are written directly after the op code to select a particular one at the 16 orders. This op code is also subject to footnote 3.

11. This code may be as a secondary operation in connection with those I/O select orders which are subject to end-of-operation interrupts.

12. These mnemonics are mathematical constants.

II. LIST OF MNEMONICS BY TYPE

G. INDEX TRANSMISSION AND ARITHMETIC

I	C + I		ADD IMMEDIATE TO COUNT
I	C - I		SUBTRACT IMMEDIATE FROM COUNT
I	KC		COMPARE COUNT
I	KCI		COMPARE COUNT IMMEDIATE
I	KV		COMPARE VALUE
I	KVI		COMPARE VALUE IMMEDIATE
I	KVNI		COMPARE VALUE NEGATIVE IMMEDIATE
I	LC		LOAD COUNT
I	LCI		LOAD COUNT IMMEDIATE
I	LR		LOAD REFILL
I	LRI		LOAD REFILL IMMEDIATE
I	LV		LOAD VALUE
I	LVE		LOAD VALUE EFFECTIVE
I	LVI		LOAD VALUE IMMEDIATE
I	LVNI		LOAD VALUE NEGATIVE IMMEDIATE
I	LVS		LOAD VALUE WITH SUM
I	LX		LOAD INDEX
I	RNX		RENAME
I	SC		STORE COUNT
I	SR		STORE REFILL
I	SV		STORE VALUE
I	SVA		STORE VAL UE IN ADDRESS
I	SX		STORE INDEX
I	V +		ADD TO VALUE
I	V + C		ADD TO VALUE AND COUNT
I	V + CR		ADD TO VALUE, COUNT AND REFILL
I	V + I	9	ADD IMMEDIATE TO VALUE
I	V+IC	9	ADD IMMEDIATE TO VALUE AND COUNT
I	V+ICR	9	ADD IMMEDIATE TO VALUE, COUNT AND REFILL
I	V -I	9	SUBTRACT IMMEDIATE FROM VALUE
I	V -IC	9	SUBTRACT IMMEDIATE FROM VALUE AND COUNT
I	V -ICR	9	SUBTRACT IMMEDIATE FROM VALUE, COUNT AND REFILL

H. SYSTEM SYMBOLS THAT ARE BIT ADDRESSES

LOCATION WORD NO.	BIT ADDRESS	SYSTEM SYMBOL	MNEMONIC	NAME
0	0-63	$	Z	WORD NUMBER ZERO
1	0-18	$	IT	INTERVAL TIMER
1	28-63	$	TC	TIME CLOCK
2	0-17	$	IA	INTERRUPTION ADDRESS
3	0-17	$	UB	UPPER BOUNDARY
3	32-49	$	LB	LOWER BOUNDARY
3	57	$	BC	BOUNDARY CONTROL
4	0 - 63	$	MB	MAINTENANCE BITS
5	12-18	$	CA	CHANNEL ADDRESS
6	0-18	$	CPU	OTHER CPU
7	17-23	$	LZC	LEFT ZEROS COUNT
7	44-50	$	AOC	ALL ONES COUNT
8	0-63	$	L	LEFT HALF OF ACCUMULATOR
9	0-63	$	R	RIGHT HALF OF ACCUMULATOR
10	0-7	$	SB	SIGN BYTE
				INDICATORS
11	0-63	$	IND	INDICATORS
11	0	$	MK	MACHINE CHECK
11	1	$	IK	INSTRUCTION CHECK
11	2	$	IJ	INSTRUCTION REJECT
11	3	$	EK	EXCHANGE CONTROL CHECK
				ATTENTION REQUEST
11	4	$	TS	TIME SIGNAL
11	5	$	CPUS	CPU SIGNAL
				INPUT-OUTPUT REJECTS
11	6	$	EKJ	EXCHANGE CHECK REJECT
11	7	$	UNRJ	UNIT NOT READY REJECT
11	8	$	CBJ	CHANNEL BUSY REJECT
				INPUT-OUTPUT STATUS
11	9	$	EPGK	EXCHANGE PROGRAM CHECK
11	10	$	UK	UNIT CHECK
11	11	$	EE	END EXCEPTION
11	12	$	EOP	END OF OPERATION
11	13	$	CS	CHANNEL SIGNAL
11	14	$		RESERVED