United Aircraft Corporation

March 22, 1956

Note: This write-up cancels and supersedes write-up dated September 21, 1955. The effect of Addendum No. 1 is included in this write-up.

SHARE Assembler

Reproduced at Lincoln Laboratory

(UA SAP 1 and 2)

Roy Nutt

704 instructions to be assembled by this program are written with references expressed as arithmetic combinations of symbols and/or decimal integers. A variable field format is used in which the parts of the instruction are given in the order address, tag and decrement. In addition to instruction; data in decimal, octal or Hollerith (BCD) form may be assembled, and library routines written in the same symbolic form may be conveniently incorporated into the program being assembled.

A program to compute

$$P_N(x,y) = \sum_{j=0}^{N} \left[\sum_{i=0}^{i+j=N} a_{ij} x^i \right] y^j$$

is used as an example. (See last page).

In order to describe the use of this assembly program, let us consider first a simplified explanation of symbolic assembly operation.

The procedure is divided into two parts; - the first examines the program to be assembled in order to define each symbol used in writing the program. The second part prepares the actual machine language program, punches it in binary form on cards and produces a printed copy of the program in symbolic form together with the corresponding octal machine language program.

During the first part a counter is used to specify the absolute location of each word in the program. Call this location counter L. L is set initially to an integer supplied to the assembly program by the program being assembled, henceforth L is increased by one for each word to be used by the program.

Simultaneous1y with this counting procedure a table is constructed. Each entry in this table defines a symbol used in the program as being equivalent to some integer. Entries to the table are made in two ways:

1. A symbol appears as the "symbolic location" of a word in the program being assembled and is assigned the value of L.

2. A symbol is defined by a pseudo operation.

It is important to note that the order of the absolute instructions preduced by symbolic assembly is determined solely by the order in which the symbolic instructions are read by the assembly program.

During the second part of the assembly process L is computed in exactly the same manner as it was during the first part. In addition all symbols in the symbolic program are replaced by the integer equivalences given in the table formed during the first part, thus producing an absolute program.

Note that this operation requires that each symbol be uniquely defined.

For use in the assembly program the following definitions are made:

Symbol: Any combination of not more than 6 Hollerith characters, none of which is + - * / , $ and at least one of which is non-numeric.

Integer: (with respect to instructions):
Any decimal integer less than 1 000 000.

The operation part of each instruction is specified by the standard "SHARE" abbreviation of 3 alphabetic characters.

A symbolic instruction should be identified by a symbol ("symbolic location" only if it is necessary to refer to this instruction in the program.

The address, tag and decrement parts of symbolic instructions are given in that order. In some cases the decrement, tag or address parts are not necessary, therefore the following combinations are permissible.

```
OP

OP    Address

OP    Address, Tag

OP    Address, Tag, Decrement
```

Examples of the last three types occur in the illustrative problem at P3, P3+2 and P3+3 respectively.

Note the tag, if present, must be separated from the address by a comma and similarly the decrement, if present, must be separated from the tag by a comma. For the few instructions which require a tag bur no address, the address zero shoul be used, for example:

```
PDX 0,4
```

Similarly where a decrement is required with no tag a zero should be used as in

```
TXL A,O,B
```

The following card form is used by the assembly

1-6	Symbol or blank
7	Blank
8-10	Abbreviated operation or blank:
11	Blank
12-72	Variable field
73-80	Not used

Expressions defineing the address, tag and decrement are punched without blanks from column 12 on. The first blank to the right of column 12 defines the end of the instruction. All punching to the right of such a blank is considered to be a remark and has no effect on the assembly process.

If an instruction requires a symbolic location, the symbol used is punched in columns 1-6.

Ar1thmetic expressions

Arithmetic expressions in terms of symbols and integers may be used with some pseudo-operations and to define address, tag or decrement parts of 704 instructions.

The following elementary operations may be used:

addition, indicated by +

subtraction, indicated by -

multiplication, indicated by *

division, indicated by /

No parenthethical expressions may be written.

Integral arithmetic modulo 2^{35} is used, hence

1. Multiplication is not commutative with division:

   A*B/C≠A/C*B

   except when C is a factor of A.
   Note that A/C*B implies (A/C)*B net A/(C*B).

2. Addition and subtraction are commutative:

   A+B-C=B-C+A

3. Multiplication and division are distributive with respect to each other but <u>not</u> with respect to addition or subtraction:

   A+B*C/D≠A*C/D+B*C/D

   Note that A+B*C/D implies A+(B*C/D) and not (A+B)*C/D

If the result of an expression is to be expressed in n binary places, its value is computed modulo 2^n. If the residue is negative, its 2's complement is the result.

Hence if v is the value of an expression, r is the result used and

$$m = |v| \mod 2^n$$

$$\text{then } r = \begin{cases} m & v \geq 0 \\ 2^n - m & v < 0 \end{cases}$$

For example the instruction at location P3+3 in the i11ustration has a decrement part of -1. Here. m=1, v=-1, n=15 so that

$r=2^{15}-1$

Consider also the tag part of instruction on P4-1 where

v=J+K=1+4=5

m=5, n=3

so that r=5

PSEUDO OPERATIONS

Origin specification: ORG

The location counter L is set to the value of the expression appearing in the variable field. Each symbol appearing in the expression must have been previously defined (i.e. appeared in the symbol field, columns 1-6, of some instruction or pseudo-instruction preceding this origin specification.)

If no origin specification is given for a program the initial value of L shall be zero.

Origin specification instructions may be used at will.

Equals: EQU

They symbol appearing in 1-6 is assigned the integer value given by the expression appearing in the variable field. Each symbol used in this expression must be previously defined.

Note that the pseudo operation EQU is to be used only in those cases where the symbol appearing in 1-6 specifies a preset program parameter such as the order of a matrix, the pegree of a polynomial, the number of items in a group, or any other quantity wheich is invariant with respect to the location of the program in storage. If the symbol specifies the location of a piece of data or an instruction, the pseudo operation SYN should be used.

Synonym: SYN

The symbol appearing in 1-6 is assigned the integer value given by the expression appearing in the variable field. Each s,ymbol used in this expression must be previously defined.

Note that the pseudo operation SYN is to be used only in those cases where the symbol appearing in 1-6 specifies the location of a pi~ce ot data, the location of an instruction, or any other quantity whose value depends upon the location ot the program in storage. If the symbol specifies a preset program parameter, the pseudo operation EQU should be used.

Decimal data: DEC

The decimal data beginning in column 12 is converted to binary and assigned to consecutive locations L, 1+1, ...

Successive words of data on a card are separated by commas, and the first blank to the right of column 12 indicates that all punching to the right of this blank is a remark.

Signs are indicated by + or - (12 or 11 punch) preceding the number, the exponent or the binary scale factor. Howerer it is not necessary to use the + sign.

The symbol appearing in 1-6 identities the first decimal data word on the card. Successive words are identified relative to the first word. If no symbol appears in 1-6, the data words are identified relative to that word most recently identified by a symbol.

If none of the characters . E or B appear in a decimal data word, the word is converted as a binary integer with the binary point at the right hand end of the word.

If either of the characters E or . or both appear in a decimal data word and the character B does not appear, the word is converted to a 704 type floating binary quantity. The decimal exponent used in this conversion is the number which follows inunediately after the character E. If the character E does not appear, the exponent is assumed to be zero. If the decimal point does not appear, it is assumed to be at the right hand end. For example, 12.345, +12.345, 1.2345El, l234.5E-2 and l2345E-3 are all equivalent representations of the same floating point word.

If the character B appears in a decimal data word, the word is converted as a fixed point binary quantity. Tne binary scale factor used in this conversion is the number which follows immediately after the character B; - it being the number of binary places between the left hand end of the storage cell and the binary point of the fixed point binary result. If the decimal point does not appear in the decimal data word, it is addumed to be at the right hand end. The decimal exponent used in this conversion is the number which follows immediately after the character E. The order of B and E is not significant. For example, 12.345B4, +1.2345E1B4 and l2345B4E-3 are all equivalent representations of the same fixed point quantity.

Octal data: OCT

The octal data beginning in column 12 is taken in binary integer form, the binary point considered to be on the right hand end of a 704 word, and assigned to consecutive storage locations L, L+l,...

Successive words are separated by commas and the first blank to the right of column 12 indicates that all punching to the right is to be considered a remark.

The symbol appearing in 1~6 identifies the first octal data word on the card. Successive words are identified relativ~ to the first word. If no symbol is used, identification is relative to that word most recently identified bya symbol.

In the case of 12 digit octal numbers, the following equivalences exist with respect to the high order digit:

$$-0 \equiv 4 \quad -1 \equiv 5 \quad -2 \equiv 6 \quad -3 \equiv 7$$

Either fom may be used in coding for the assembly.

Hollerith data: BCD

Nomally the 10 six character wc:rrds of Hollerith information from col. 13-72 are read and alSsigned to locations L, L+l,...,L+9. If, however, less than 10 BCD words are desired, a word count v ($0<v<9$) is punched in colwnn 12, in which case v words are read and assigned to locations L, L+l,...,L+v-l.

The symbol appearing in 1-6 identifies the first Hollerith word on the card. Successive words are identified relative to the first word. If no symbol is used, identification is relative to that word most recently identified by a symbol.

Block started by symbol: BSS

The block of storage extending from L to L+N-1, where N is the value of the expression beginning in column 12, is reserved by this pseudo operation. Each symbol in the expression for N must have been previously defined.

If a symbol is punched in 1-6, it is assigned the value L, corresponding to the first word of the block reserved.

Finally, L is replaced by L+N.

Block ended by symbol: BES

This pseudo operation is exactly the same as BSS, except that the value assigned. to any symbol appearing in 1-6 is L+N, corresponding to the location of the first word following the block reserved.

Repeat: REP

Two expressions, the first beginning in column 12 and separated from the second by a comma, define two integers M and N. The block of instructions and/or data preceding the REP operation in locations L, L+l,...,L+M-l is repeated N times, the repeated infomation being assigned to locations L+M, L+M+l,...,L+M*N-l. Only one word of infomation may appear on each card which is part of a repeated block.

Library search: LIB

The library routine identified by the symbol in 1-6 is obtained trom a library tape and inserted in the program being assembled. If the library routine requries k words of storage it will occupy 1ocations L, L+l,...,L+K-l. The identification symbol is not entered in the table of symbols, but any symbols appearing in the library routine are entered and properly defined.

The first set of information on the library tape is an ordered list of the subroutines which are on the tape. The assembly program always keeps track of the position of the library tape and makes use of the information in the ordered list of subroutines in order to search the library in the most efficient manner. (The library tape is not rewound between searches.)

Tape searching time may be minimized both by recording the most frequently used subroutines at the beginning of the tape and by specifying that the subroutines to be incorporated into any particular program are called for in the order in which they appear on the tape.

Heading: HED

It is often convenient to combine several programs into one program. Two difficulties immediately arise. First, the symbolic references to data common to the seyeral programs may differ in the individual programs. This can be easily corrected by the use of synonyms which equate the proper symbols.

Second, it may be that two or more of the individual programs use the same symbols for references which should be unique. In order to restore uniqueness, it is necessary to change the symbols in each program in some way. The heading pseudo operation accomplishes this result in the following manner.

The heading card supplies to the assembly program a single character (punched in column 1 of the HED card). Each symbol in the program following the HED pseudo operation is prefixed by this character except when a special indication to cancel the prefixing operation is given. A new heading pseudo operation will replace the prefix character. Thus several programs having non-unique symbols may be combined by giving the heading pseudo operation with a unique character before each program. If a numerical heading is used, then some non-numeric character must be punched in 2-6 of the heading card.

It is, however, sometimes necessary to make cross-references between the individual programs. To accomplish this, such references must be written in the following way. Let H be the heading character and K be the symbol in the block headed by H to which reference is to be made. To refer to K from a part of the program not headed by H write

H$K

The special character $ indicates to the assembly program that K is to be prefixed by H instead of by the prefix given on the last heading card.

It is important to note that if use is to be made of the Heading feature, all symbols used throughout the program will usually be restricted to five or fewer characters. If any six-character symbols (such as the erasable storage designation COMMON) are used, these symbols will <u>not</u> be headed.

<u>Define: DEF</u>

If there exist in the program symbols not defined in accordance with the normal rules, such symbols may be defined in a different manner by use of the pseudo operation DEF. This pseudo operation causes the first such symbol encountered in an address, tag or decrement to be assigned the value given by the expression betinning in column 12 of the DEF card, Successive undefined symbols are then given successive values until either a new DEF is given (in which case a new assignment is begun) or until the capacity of the symbol table is exceeded.

Note that the pseudo operation DEF can not be used tode.f1ne an otherwise undefined symbol if this symbol occurs in the address, tag or decrement of an instruction which precedes the DEF card. The pseudo operation DEF defines only those otherwise undefined symbols which are first encountered after the DEF card itself has been encountered.

Similarly, if two DEF cards are used, and if an otherwise indefined symbol occurs both in instructions which appear between the two DEF cards, as well as in instructions which follow the second DEF card, then the definition which will be used throughout is the one established by the first DEF card. The secondDEF card has in such a case no effect on the already - established definition.

Remarks: REM

Any Hollerith punching in column 12-72 will be reproduced in the printed listing of the assembly, without otherwise affecting the assembly in any way.

End of program: END

This pseudo operation must be the last read by the assembly program. The value of the expression beiinning in column 12 is punched as the transfer address in a 704 binary correction/transfer card.

Operation code

In addition to the standard 3 letter operation code adopted by SHARE, this assembly program recognizes the following codes which may be used to assign arbitrary values to the prefix and sign of calling sequence words:

Alphabetic Code	Name	Octal Code
MZE	Minus zero	-0000
MON	Minus one	-1000
MTW	Minus two	-2000
MTH	Minus three	-3000
PZE	Plus zero	+OOOO
PON	Plus one	+1000
PTW	Plus two	+2000
PTH	Plus three	+3000
FOR	Four	-0000
FVE	Five	-1000
Six	Six	-2000
SVN	Seven	-3000

In coding symbolic instructions which have OFF, CHS, CLM, COM, DCT, ETM, IOD, LTM, LBT, PBT, RCD, RPR, RTT, RND, SLF, SPT, SSM, SSP, WTV, WPR, or WPU as their operation part, the address part should be blank or zero, since the assembly program automatically introcduces the correct address.

In coding symbolic instructions which have BST, RDR, RTB, RTD, REW, SLN, SLT, SPR, SPU, SWT, WDR, WEF, WTB, WTD, or WTS as their operation part, the address part should be the unit number (in decimal). For instance, BST 2 implies Back Space Tape No. 2, SPR 9 implies Sense Printer Exit No. 9, WDR 3 implies Write Drum No. 3, and so on. The assembly program automatically computes the correct octal address (222, 371, and 303 respectively, in the foregoing examples).

Location counter

If an absolute decimal location (i.e.: one containing no non-numeric characters) is punched in columns 1-6 of any card in the assembly, the location counter L will be set to that value. The effect of absolute decimal punching in these columns is therefore identically the same as if the card in question were to be placed immediately behind an ORG card having the exact same absolute decimal location punched in its variable field.

Operational features

As an aid to the programmer this assembly program gives some indications of erroneously prepared programs.

If a symbol used in the program is not defined; address, tag or decrement parts containing this symbol are left blank in the printed assembly and zero is used for the corresponding address, tag or decrement parts in the binary instruction deck. In the case of pseudo operations involving undefined symbols, any expressions containing such symbols are evaluated using zero as the value of the undefined symbol. A list of all undefined eymqols will be printed at the end of the assembly. (Included in this list will also be any symbols which have been defined by means of the pseudo operation DEF).

If a non-existent operation code is used, the prefix part of the corresponding instruction is left blank in the printed assembly and zero is used as the operation code in the binary deck.

A list of duplicated symbols is printed prior to the printing of the program. This list gives the symbol duplicated and the integer values assigned to it.

Other convenient features are:

Printing of the entire program may be suppressed, or printing of the subroutines copied from the library may be suppressed.

Single or double spacing is optional.

Assembly may be made from either a BCD tape, or from cards.

Binary punching is available in either non-relocatable or relocatable format.

Capacity of the symbol table

Sufficient space has been set aside in core storage in order to permit the assembler to construct a symbol table containing 1097 entries. In cases where the program to be assembled makes use of the library tape, however, the maxinum number of symbols which the assembler can handle is somewhat reduced. This follows from the fact that the entire ordered list of subroutines which forms the firat set of information on the library tape is copied into the upper end of the symbol table area at the time that the first LIB card is encountered. Hence if the library tape is used during an assembly, the effective symbol table size becomes 1097 minus the number of library subroutines on the tape.

In connection with the capacity of the symbol table, it should also be noted that any unassigned symbols are also recorded in this symbol table area preparatory to printing the list of unassigned symbols at the end of the assembly. Hence if a case arises where the number of assigned symbols plus the number ot subroutines in the tape library (if used) plus the number of unassigned symbo1s should total more than 1097, then the list of unassigned symbols printed at the end of the assembly will include only enough symbols to make up the 1097 total. The rest of the unassigned symbols can only be detected by noting blank addresses, tags or decrements in the printed output.

Reassembly features

Additions to a program which has been assembled are easily accomplished if the table of symbols which was punched during the initial assembly process has been saved. It is then necessary only to reload this table and assemble the new parts of the program. The original program need not be reloaded.

Furthermore any change to the original program which does not involve relocation of any part of the program, or any reassignment of symbols, may be made by assembly of only those parts of the program which are to be changed.

Enlarged core storage

The assembler has been so written as to permit it to be used, without change, in 704's with enlarged core storage.

For each additional two words of core storage beyond the minimum of 4096, the assembler automatically provides for one &dditional syabol in the symbol table.

```
                 04000       ORG 2048
04000 -0 53400 5 04011       LXD P1, J+K        Initialize Index Registers
04001 -0 63400 4 04020 P4    SXD P2,K           Store K
04002  0 5000O 1 04022       CLA A+l,J          Obtain First Element
04003  1 77777 1 04004       TXI P6,J,-l        X
04004 -2 00001 4 04017 P6    TNX P5,K,1         X
04005  0 76500 0 00043 P3    LRS 3,             Form Polynomial
04006  0 26000 0 04046       FMP X              In X
04007  0 30000 1 04022       FAD A+l,J          X
04010  1 77777 1 04011       TXI Pl,J,-l        Step Coefficient
04011  2 00001 4 04005 P1    TIX P3,K,1         Test Reduced K
04012  0 60100 0 04051       STO S              Store Partial Sum
04013  0 56000 0 04050       LDQ Z              Form Polynomial
040l4  0 26000 0 04047       FMP Y              In Y
04015  0 30000 0 04051       FAD S              X
04016 -3 77754 1             TXL OUT,J.-R/2+1   X
04017  0 60100 0 04050 P5    STO Z              X
04020  1 00000 4 04001 P2    TXI P4,K           X
                 00005 N     EQU 5
                 00052 R     EQU N*N+3*N+2          Note that the a/sij/S's are
                 04021 A     BSS R/2                stored in the order a/s05/S,
04046  0 00000 0 00000 X                            a14′ a04′ a23′ a13′ a03′
04047  0 00000 0 00000 Y                            ...., a00 from location
04050  0 00000 0 00000 Z                            A on.
0405l  0 00000 0 00000 S
                 00001 J      EQU 1
                 00004 K      EQU 4
                 04000        END P4-1
                 00000 OUT
```

United Aircraft Corporation

March 7, 1956

SHARE Assembler Operator's Notes

(UA SAP 1 and 2)

Roy Nutt

Control panel requirements:

SHARE 72 column reader panel (1-72)
SHARE 72 column punch panel (1-72)
SHARE 72-120 printer panel

Tapes:

If the library tape and both the off-line printer and the off-line reader are used, three tapes are required, namely

logical tape 1: input tape
logical tape 2: output tape
logical tape 3: library tape

Logical tape 1 is normally needed for either off-line or on-line input. However it is not required if the symbolic deck is read twice by the on-line reader.

Logical tape 2 is not needed if off-line output is not required.

Logical tape 3 is not needed if the library is not used.

Sense switches:

1 Up and 2 up: Input to both passes is from logical tape 1 (prepared previously on the off-line reader or by an off-line reader simulator).

1 Down and 2 up: Input to the first pass is from symbolic cards read on-line. Input to the second pass is from logical tape 1 which (with this sense switch setting) is written during the t1rst pass.

1 Down and 2 Down: Input to both passes is from symbolic cards read on-line. With this sense switch setting, logical tape 1 is not used.

3 Up: Suppress on-line printing.

3 Down: Output is printed on-line.

4 Up: Any on-line printing is single spaced.

4 Down: Any on-line printing is double spaced.

5 Up: Logical tape 2 is written during the second pass, in order to permit later off-line printing.

5 Down: Suppress preparation of logical tape 2.

6 Up: Suppress printing of library subroutines.

6 Down: Library subroutines taken from the library tape are printed.

Program control:

If necessary, mount the library tape as logical tape 3.

Set the sense switches as desired.

If the symbolic deck has been written on tape, mount this tape as logical tape 1 and load UA SAP 1 followed by UA SAP 2.

If the symbolic deck is to be read on-line, place it between UA SAP I and UA SAP 2.

It the symbolic deck is to be read again during the second pass, place it also between the transfer card of UA SAP 2 and the two blank cards which follow.

UA SAP I Stops:

$(437)_8$ HTR $(400)_8$: Should occur only during on-line reading of symbolic cards (1. e.: sense switch 1 down). Indicates a card punched with an illegal (non-Hollerith) character, or a machine error while reading symbolic cards. Ready the correct card in the reader and press start. The card for which the stop occurred will be the third card back in the stacker after running the cards out of the feed.

$(1401)_8$ HTR $(1372)_8$: End of file condition while reading symbolic cards on-line. Ready remainder of deck and press start.

$(1414)_8$ HTR $(1372)_8$: End of file condition from tape. Machine error or no END card written on tape. Reassemble.

$(1543)_8$ HTR $(1375)_8$: Symbol Table has been filled. Assemble program in smaller sections.

$(1641)_8$ HTR $(1550)_8$: Library search failed twice. Press start to try again.

$(2121)_8$ HTR $(2116)_8$: Wrong board in printer. Replace with SHARE 72-120 board and press start.

$(2225)_8$ HTR $(2330)_8$: Check sum for symbol table is wrong. Probably machine error. Reassemble or press start to continue.

UA SAP 2 Stops:

$(437)_8$ HTR $(400)_8$ Should occur only during on-line reading of symbolic cards (1. e.: sense switch 2 down). Indicates a card punched with an illegal (non-Hollerith) character, or a machine error. Ready the correct card in the reader and press start.

$(1463)_8$ HTR $(1454)_8$: End of file condition while reading symbolic cards on-line. Ready remainder of deck in reader and press start.

$(1474)_8$ HTR $(1454)_8$: End of file condition while reading tape. Either machine error or no END card on tape. Reassemble.

$(2466)_8$ HTR $(2375)_8$: Library search failed twice. Press start to try again.

UA SAP 2 Transfer cards:

UA SAP 2 58 is the normal transfer card. It is used when standard SHARE absolute binary cards are to be punched.

If it is desired to punch relocatable binary cards, it is necessary to replace the transfer card of UA SAP 2 by a correction/transfer card which modifies UA SAP 2 appropriately. This card is designated

UA SAP 2 58 (UA SAP 2 NQ)

If it is desired to punch 24 words/card it is necessary to replace the transfer card of UA SAP 2 by a correction/transfer card which modifies UA SAP 2 appropriately. This card is designated

UA SAP 2 58 (UA SAP 2 EH)

Use of the symbol table:

If it is desired to assemble a symbolic program which makes reference to symbols defined by a previous assembly and the program to be assembled defines no symbols defined by the previous assembly (or in the case of duplicate definitions the duplicated symbols are defined as the same absolute number) then the symbol table from the previous assembly may be used to provide the necessary references.

To accomplish this the symbol table is loaded immediately before the transfer card of UA SAP 1 and assembly is performed in the normal way.

March 14, 1958

Op code For UASAP1

ACL	ADD AND CARRY LOGICAL WORD
ADD	ADD
ADM	ADD MAGNITUDE
ALS	ACCUMULATOR LEFT SHIFT
ANA	AND TO ACCUMULATOR
ANS	AND TO STORAGE
ARS	ACCUMULATOR RIGHT SHIFT
BST	BACKSPACE TAPE
CAC	COPY AND ADD AND CARRY LOGICAL
CAD	COPY AND ADD AND CARRY LOGICAL WORD
CAL	CLEAR AND ADD LOGICAL WORD
CAS	COMPARE ACCUMULATOR WITH STORAGE
CFF	CHANGE FILM FRAME
CHS	CHANGE SIGN
CLA	CLEAR AND ADD
CLM	CLEAR MAGNITUDE
CLS	CLEAR AND SUBTRACT
COM	COMPLEMENT MAGNITUDE
CPY	COPY OR SKIP
DCT	DIVIDE CHECK TEST
DVH	DIVIDE OR HALT
DVP	DIVIDE OR PROCEED
ETM	ENTER TRAPPING MODE
FAD	FLOATING ADD
FDH	FLOATING DIVIDE OR HALT
FDP	FLOATING DIVIDE OR PROCEED
FMP	FLOATING MULTIPLY
FOR	FOUR
FSB	FLOATING SUBTRACT
FVE	FIVE
HPR	HALT AND PROCEED
HTR	HALT AND TRANSFER
IOD	INPUT - OUTPUT DELAY
LBT	LOW ORDER BIT TEST
LDA	LOCATE DRUM ADDRESS

LDQ	LOAD MQ
LGL	LOGICAL LEFT
LLS	LONG LEFT SHIFT
LRS	LONG RIGHT SHIFT
LTM	LEAVE TRAPPING MODE
LXA	LOAD INDEX FROM ADDRESS
LXD	LOAD INDEX FROM DECREMENT
MON	MINUS ONE
MPR	MULTIPLY AND ROUND
MPY	MULTIPLY
MSE	MINUS SENSE
MTH	MINUS THREE
MTW	MINUS TWO
MZE	MINUS ZERO
NOP	NO OPERATION
ORA	OR TO ACCUMULATOR
ORS	OR TO STORAGE
PAX	PLACE ADDRESS IN INDEX
PBT	P BIT TEST
PDX	PLACE DECREMENT IN INDEX
PON	PLUS ONE
PSE	PLUS SENSE
PTH	PLUS THREE
PTW	PLUS TWO
PXD	PLACE INDEX IN DECREMENT
PZE	PLUS ZERO
RCD	READ CARD READER
RDR	READ DRUM
RDS	READ SELECT
REW	REWIND
RND	ROUND
RPR	READ PRINTER
RQL	ROTATE MQ LEFT
RTB	READ TAPE, BINARY

RTD	READ TAPE, DECIMAL
RTT	REDUNDANCY TAPE TEST
SBM	SUBTRACT MAGNITUDE
SIX	SIX
SLF	SENSE LIGHTS OFF
SLN	SENSE LIGHT ON
SLQ	STORE LEFT HALF MQ
SLT	SENSE LIGHT TEST
SLW	STORE LOGICAL WORD
SPR	SENSE PRINTER
SPT	SENSE PRINTER TEST
SPU	SENSE PUNCH
SSM	SET SIGN MINUS
SSP	SET SIGN PLUS
STA	STORE ADDRESS
STD	STORE DECREMENT
STO	STORE
STP	STORE PREFIX
STQ	STORE MQ
STZ	STORE ZERO
SUB	SUBTRACT
SVN	SEVEN
SXD	STORE INDEX IN DECREMENT
TIX	TRANSFER ON INDEX
TLQ	TRANSFER ON LOW MQ
TMI	TRANSFER ON MINUS
TNO	TRANSFER ON NO OVERFLOW
TNX	TRANSFER ON NO INDEX
TNZ	TRANSFER ON NO ZERO
TOV	TRANSFER ON OVERFLOW
TPL	TRANSFER ON PLUS
TQO	TRANSFER ON MQ OVERFLOW
TQP	TRANSFER ON MQ PLUS
TRA	TRANSFER

TSX	TRANSFER AND SET INDEX
TTR	TRAP TRANSFER
TXH	TRANSFER ON INDEX HIGH
TXI	TRANSFER WITH INDEX INCREMENTED
TXL	TRANSFER ON INDEX LOW OR EQUAL
TZE	TRANSFER ON ZERO
UFA	UNNORMALIZED FLOATING ADD
UFM	UNNORMALIZED FLOATING MULTIPLY
UFS	UNNORMALIZED FLOATING SUBTRACT
WDR	WRITE DRUM
WEF	WRITE END OF FILE
WPR	WRITE PRINTER
WPU	WRITE PUNCH
WRS	WRITE SELECT
WTB	WRITE TAPE, BINARY
WTD	WRITE TAPE, DECIMAL
WTS	WRITE TAPE SIMULTANEOUSLY
WTV	WRITE CRT
WTV	WRITE CRT