UCRL-8932

UNIVERSITY OF CALIFORNIA
Lawrence Radiation Laboratory
Berkeley, California

Contract No. W-7405-eng-48

IBM MODEL-704 GUIDEBOOK

Donald H. Zurlinden

October 5, 1959

Printed for the U. S. Atomic Energy Commission

DISCLAIMER

IBM MODEL-704 GUIDEBOOK

Contents

IBM MODEL-704 GUIDEBOOK

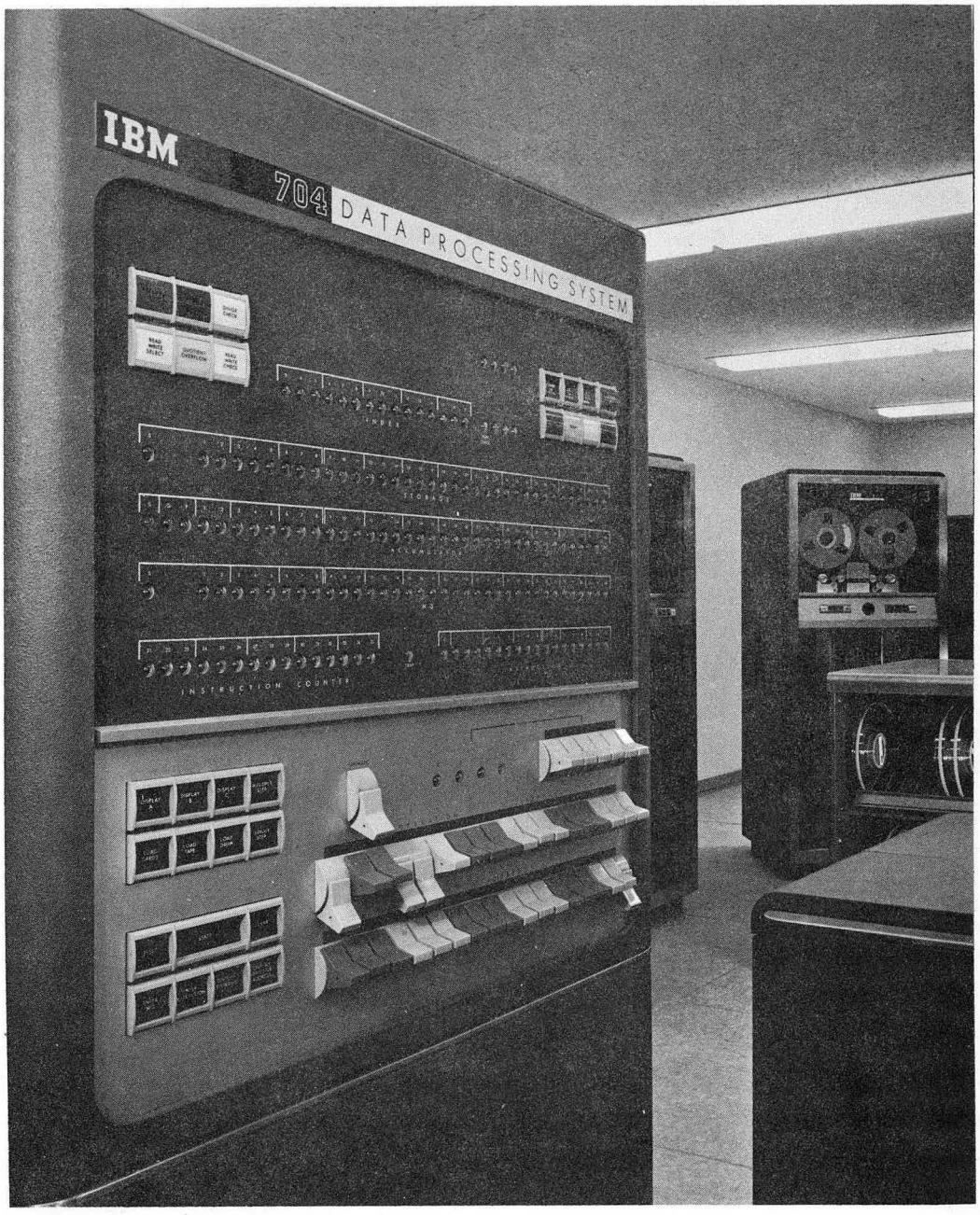
Donald H. Zurlinden

Lawrence Radiation Laboratory
University of California
Berkeley, California

October 5, 1959

ABSTRACT

An IBM-704 Guidebook for LRL users of the Computer Center is presented. This book describes the material on systems available at the Computer Center and the operational procedures that will be followed. Sample problems are included. The Guidebook is to be used in conjunction with existing material on the IBM-704 computer and its applications.



ZN-2243

The IBM-704 data-processing system.

IBM MODEL-704 GUIDEBOOK

Donald H. Zurlinden

Lawrence Radiation Laboratory
University of California
Berkeley, California

October 5, 1959

INTRODUCTION

The IBM 704 Guidebook is divided into three sections: I Configuration of the IBM 704 Facility, II The Share Assembler, and III The Fortran-II System. It is assumed that the reader understands basic computing methods and is familiar with the components of the IBM-704 and IBM auxiliary equipment. Section I is self-explanatory. Section II contains sufficient information to enable the programmer to complete a problem from coding through the assembly of a working binary program. The list of library programs and their abstracts are not stated in this Guidebook. The purpose of Section III, The Fortran-II System, is to describe the basic information required to compile and run a written source program. The material required to write a Fortran source program is found in:

(a) Reference Manual, Fortran Automatic Coding System, C28-6003, October 1958.

(b) Reference Manual, Fortran II, C28-6000-1, December 1958.

(c) Programmer's Primer for Fortran Automatic Coding System, 32-0306-1, March 1958.

Material used in the Guidebook came from the above sources as well as from the following:

(a) Share Reference Manual for the IBM 704, August 1956.

(b) The Share Assembler, UA SAP 3-7, (received through the Share organization).

The Guidebook will continue to develop and increase in content, and therefore it was written and arranged so that new sections may be added or existing sections may be revised. Criticisms and suggestions which will help improve the Guidebook are solicited.

I. COMPONENTS OF THE IBM FACILITY

The components of the IBM-704 facility are:

IBM No.	Description
704	Central processing unit
738	Magnetic-core storage unit (32768 words)
727	Magnetic-tape units (8 total)
753	Tape-control unit
711	Punched-card reader (250 cards per minute)
721	Punched-card recorder (100 cards per minute)
716	Alpha-numeric printer (150 lines per minute, 120 characters per line)
736	Power Frame No. 1
741	Power Frame No. 2
746	Power-Distribution Unit
Code 419	Floating-Point Trap
704 Inst.	Copy and add logical work
717	Peripheral printer (150 lines per minute, 120 characters per line)
757	Peripheral control unit

The off-line printer will use one of the eight tape units listed above.

II. THE SHARE ASSEMBLER

A. Card and Tape Formats

Input for conversion to instruction is punched in symbolic format. The Share Assembler converts the symbolic input to machine language and punches binary output cards. The format of the binary output cards will be either absolute or relocatable. Generally a program will be punched in absolute format, since relocatable cards are used only when the program must be relocated each time it is read into the IBM-704 machine. The data from the binary cards are usually read into the computer by specified loaders. The input to the assembly is always in symbolic card form except in the case of the symbol table.

All cards (on-line or off-line) that contain 72 columns of information and eight columns of identification are to be punched with the information in Columns 1 to 72 and the identification in Columns 73 to 80.

1. Symbolic Cards

The card format used by UA SAP has the following form:

Column(s)	Description
1 to 6	Location field or blank
7	Blank
8 to 10	Operation code or blank
11	Blank
12 to 72	Variable field
73 to 80	Not used.

All punching is in Hollerith Code. Expressions defining the address, tag, and decrement portion of a variable field are punched without blanks, beginning with Column 12. The variable field extends to the first blank to the right of Column 12. A variable field which does not begin by Column 12 is assumed to be blank. All punching to the right of the first blank of a variable field is considered to be a remark and has no effect on the assembly process. If an instruction does not have a variable field (CHS, SSP, SSM, etc.), remarks or comments may not begin before Column 13.

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

2. Binary Cards

A binary card is divided into 12 rows with Columns 1 to 36 of Row 9 designated as word 9L, Columns 37 to 72 of Row 9 designated as word 9R, Columns 1 to 36 of Row 8 designated as word 8L, etc. Hence a maximum of 24 binary words of information may be punched in one card.

The positions of a 36-bit binary word are designated from left to right as

sign 1, 2, 3, ..., 35.

For convenience, we shall use an abbreviated designation for various parts of the card and a word. P, D, T, and A stand respectively for the prefix, decrement, tag, and address of a word. The sign bit is denoted by S. Then for example, the thirteenth bit position of the word in the left half of the sixth row would be denoted by 6L13. The decrement field of the same word would be 6LD. The 9L word is always the control word, and the 9R word is always the 36-bit ACL check sum (denoted by CKS).

a. Absolute data. Bits 9L13 to 9L17 contain the word count V; 9L21 to 9L35 contain the initial location R. All other positions in 9L are ordinarily blank. Words 8L, 8R, 7L, 7R, ... contain the absolute data. The maximum word count is $22_{10} = 26_8$. If 9L2 is punched, the CKS should be ignored and no check should be made against it. This applies also to a completely blank CKS.

b. Relocatable data. Relocatable binary cards contain data whose core location is determined each time the data values are read into the 704. The necessary controls are given by the origin table. The relocatable data are usually instructions that may contain both absolute and relocatable values.

Position 9L1 is punched. Positions 9L13 to 9L17 contain the word count V; 9L21 to 9L35 contain the nominal initial location R. All other positions in 9L are ordinarily blank. If 9L2 is punched, the CKS is to be ignored, as in the case of a completely blank CKS. The indicator bits are in row 8, starting from the left. The following one- and two-bit codes are used to indicate the type of field:

0	absolute field
10	relocatable direct field
11	relocatable complemented field.

"Direct" here means uncomplemented. Relocation complemented to the address or decrement fields results in obtaining the two's complement of the field before reference to the table, and again the two's complement is taken before storing. The string of these codes starts at 8LS and proceeds continuously to the right until it terminates. Words 7L, 7R, 6L, 6R, ... contain the relocatable data. Let us, for illustration, suppose that 7LD is absolute, 7LA is relocatable direct, 7RD is absolute, 7RA is relocatable and complemented, 6LD is relocatable direct, 6LA is absolute, 6RD is absolute, and 6RA is relocatable complemented. The indicator bit pattern would be

0 10 0 11 10 0 0 11.

This may be condensed into

010011100011 ,

and this pattern is to be punched into row 8 beginning with 8LS.

c. Correction and (or) transfer cards.

(1) Correction. Rows 8 through 12 contain corrections which are entered in the following manner: The nominal location is punched in the LA field and the correction word itself in the right-hand word of the same row. If the location is to be adjusted by an increment (i. e., the correction word is to be relocated), then the L1 bit is punched. (Note that the L1 bit always indicates relocation). If a row is completely blank, it is ignored. The indicator bits for the decrement and address fields of the correction word are punched in the L3-to-L6 bit positions, according to the indicator scheme outlined in Section (b). The sequence of correction entries is assumed to be from row 8 upward. If the L20 bit (LT3) is punched, then the nominal location is assumed to be one more than the preceding one. Hence, it is not necessary to punch every nominal location in a consecutive block. If this punch (L20) appears in row 8, however, it means that the nominal location is the one actually punched in 8LA. Hence it is possible to load absolute zero at location zero. An L2 punch causes the correction in its row to be ignored. If 9L2 is punched, the CKS is ignored. No punches need appear in the 9L word, or in the 9R word, if there is to be no CKS comparison.

(2) Transfer. The contents of the 9LA field are taken to be the location to which control is to be transferred after all corrections have been loaded. If 9L1 is punched, then this nominal location is to be relocated in the usual manner. A 9L2 punch suppresses transfer as well as CKS checking.

d. Origin table. This card precedes a relocatable binary deck and it is used to define the locations required to store the program. Bit 9L12 is punched. If 9L2 is punched, the CKS is ignored as usual. Starting with row 8, the card contains a table of origins in the following format.

In each row, the nominal location that begins a region is punched in the LA field. The operating location (i. e., the final location of an instruction when it is actually to be executed) is punched in the RA field. If there is a loading location distinct from the operating location, this is punched in the RD field. If there is no loading location, then the operating location is used in place of it. The entries need not be punched in order of ascending nominal locations. If a row is completely blank or if the L2 bit is punched in an otherwise blank row, then nominal zero will be set to absolute zero.

The difference between the nominal and reference locations is computed when the origin table is stored. Table look-ups are performed on each instruction to find the blocks of storage into which the nominal location, address, and decrement fall. Then each part of the instruction is modified (subject to the indicator bits) according to the difference associated with the block into which it falls. A general binary loader that fulfills these specifications is PK CSBA.

3. Tape Format

All tape records representing 80-column cards shall be 84 characters in length, and the last form characters shall be blank. This is in regard to peripheral equipment not in use at this time.

B. United Aircraft Symbol Assembly Program

1. Description

IBM-704 instructions to be assembled by this program are written with references expressed as arithmetic combinations of symbols and (or) decimal integers.

The input to the assembler is punched on cards in a fixed-variable field format. The first field is fixed and contains the symbolic location of the instruction. The second field is also fixed and contains the mnemonic three-letter code for the instruction. The third field is a variable-length field in which the parts of the instruction are given in the order: address, tag, and decrement. In addition there is a class of pseudo-operations that look much like 704 symbolic instructions but are used to control the assembler itself. Among these are operations that permit the assembly of data [decimal, octal, or Hollerith (BCD)], the convenient incorporation of subroutines from standard and (or) local libraries, control of binary output format, and a number of other functions.

A few examples are included to help illustrate the basic uses of the pseudo-operations and the use of the assembler.

The actual assembly procedure is divided into three parts. The first two are called the first pass and second pass. The third is called the analysis section. In the first pass, the entire set of input that constitutes a single assembly is examined sequentially in order to define (i.e. assign an absolute numerical value to) each symbol that occurs in the symbol field of a 704 instruction or of certain pseudo-operations. The second pass prepares the actual machine-language program, punches it in binary form on cards and produced a printed copy of the program in the original symbolic form together with the corresponding printed, octal, machine-language program. The analysis section not only seeks out and prints a list of undefined symbols and a list of multiply defined symbols but also prints and punches certain statistics (i.e., number of symbols, tape errors, etc.)

During the first pass, a counter, called a location counter, is used to specify the absolute location of each word in the program. The location counter is increased by one for each word to be used by the program. In addition certain pseudo-operations have the ability to interrupt this unitary,

monotonic increase and either set the location counter to an arbitrary value or to increase or decrease the counter by some arbitrary value.

Simultaneously with the above modification of the location counter, a table is constructed. Each entry in this table defines a symbol appearing in the symbol (first) field as being equivalent to some integer. Entries are made in two distinct ways:

(a) A symbol appears as the symbolic location of a word in the program being assembled and is assigned the current value of the location counter.

(b) A symbol appearing in the symbol field of one of certain pseudo-operations is assigned a value that is a function of the variable field of the pseudo-operation and, sometimes, of the current value of the location counter also.

It is important to note that, except in the immediate presence of certain location counter modifiers (pseudo-operations), the location and therefore order of a given instruction produced by the symbolic assembler is determined solely by the order in which the symbolic instructions are read by the assembly program (i. e. by the physical order of the symbolic cards).

During the second pass, the location counter is computed in exactly the same manner as it was during the first pass. In addition, all symbols are replaced by the integer equivalences given in the table formed during the first pass. Thus an absolute binary program is produced. Note that this operation requires that each symbol be uniquely defined. For the purposes of the assembly program, symbol and integer are defined thus:

Symbol: Any combination of not more than six (6) Hollerith characters, none of which is plus (12 punch), minus (11 punch), asterisk, slash, comma, dollar sign or blank and at least one of which is nonnumeric. (For this purpose equals (8-3 punch) and dash (8-4 punch) are numeric.)

Integer: (with respect to variable field parts): Any decimal integer less than 1,000,000.

The operational part of each instruction is specified by the standard SHARE mnemonic of three alphabetic characters.

Ordinarily an instruction or storage-cell location should be identified by a symbolic location only if it is necessary to refer to this location 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, where OP represents the instruction abbreviation, are permissible:

OP
OP Address
OP Address, Tag
OP Address, Tag, Decrement
OP , Tag
OP , Tag, Decrement
OP ,, Decrement
OP Address,, Decrement.

Note that 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. Where no character appears between the beginning of the field and a comma, or between comas, or between the last comma and the end of the field (a blank), that part of the field is taken to be zero.

The following card format is used by the assembly program:

Columns	Contents
1 to 6	Symbol, absolute decimal location or blank
7	Blank
8 to 10	Mnemonic operation code or blank
11	Blank
12 to 72	Variable field

Expressions defining the address, tag, and decrement are punched without blanks from Column 12 on. The first blank (and there must be at least one 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. Note, however, that even characters which are not directly involved in the assembly process must be legal Hollerith characters whether the symbolic deck is to be read into the 704 on-line or off-line.

If an instruction requires a symbolic location, the symbol is punched in Columns 1 to 6. Blanks and leading zeros are not significant. That is, if X denotes blank and 0 is a zero, the following symbols are the same:

XXXXAB
XXXAXB
XABXXX
0000AB
XX00AB
XX0AXB, etc.

a. Arithmetic expressions. As stated before, the references that may be used can consist of arithmetic expressions of symbols and (or) decimal integers. The arithmetic operations allowed in these expressions are addition, subtraction, multiplication, and division designated by + (12 punch), - (11 punch), *, and /, respectively. However, no parentheses are allowed.

All of the arithmetic operations are done with 35 binary-place integral arithmetic (i. e. modulo 2^{35}). In the case of division, only the integral quotient is retained. The remainder is discarded. The evaluation of an arithmetic expression then proceeds as follows: Each segment of the expression is separately evaluated from left to right, where a segment is that portion of the expression from a + or - sign (or the beginning of the expression) to the next + or - sign (or the end of the expression) and the consecutive multiplications and divisions are performed as specified. As each segment of the expression is evaluated, it is combined from left to right as indicated by the connective + or - signs.

As an example of the above procedure, the expression

A + 200/15/6 * 15 - B/C * D

is taken to have the meaning

$$A + \left[\left[\frac{200}{15}\right] \Big/ 6\right] * 15 - \frac{B}{C} * D$$

where the brackets denote "integral part of".

Finally, if the result of an expression is to be expressed in n binary places, its magnitude is computed modulo 2^n. This quantity is taken to be the result unless the expression is negative, in which case the 2's complement is taken as the result. Hence, if v is the value of an

expression

$$m = |v| \pmod{2^n},$$

we obtain for the result,

$$r = \begin{cases} m & \text{for } v \geqslant 0 \\ 2^n - m & \text{for } v < 0. \end{cases}$$

Consider the instruction TIX P1, J+K, -1 for J = 1, K = 4. Then for the decrement part we have v = -1, m = 1, n = 15, and therefore $r = 2^{15} - 1$.

For the tag part, we have v = J + K = 1 + 4 = 5, m = 5, n = 3, and thus

$$v = 5.$$

The decimal integers that are allowed in constructing expressions are limited to values less than 1,000,000. The symbol values that are allowed are those integers less than 2^{15}, all larger integers being taken modulo 2^{15}.

If a symbol is given a negative value, it is "negative" in the sense that the assembler uses the 2's complement (modulo 2^{15}) of the magnitude of the value. For example, if one states that N = -6, then the value the assembler will use for the symbol N will be $(2^{15} - 6)$.

b. Access to the location counter. Besides being used for the multiplication sign in variable-field arithmetic, the asterisk (*) may be used to obtain the current value of the location counter. This facility allows one to avoid the use of dummy symbols. For example,

```
TRA * -5
```

is taken to mean transfer to that location which is five less than the location at which the transfer is stored.

Because the asterisk is also used to denote multiply, the following relationships hold in the variable field:

Expression	Equivalence
*	C (the location counter)
**	0
***	C (the location counter)^2
-	0
/	1
A*	0
*A	Undefined

If an asterisk begins the variable field and is followed by a variable-field operator (i. e. +, *, blank etc.) other than $ (the heading operator) or if an asterisk is preceded and followed by such an operator, the current contents of the location counter is substituted for the asterisk. Otherwise the asterisk is taken to mean multiply. The example *A is the case where the asterisk follows a variable-field operator but is not followed by such an operator. Because of the way the variable field is scanned, 0*A looks exactly equal to zero.

c. Location Counter. If an absolute decimal number is punched in the symbol field of any card in the assembly other than ABS, BOS, C/T, DEF, END, EQU, FIN, FUL, HED, LIB, LOC, OPD, ORG, PLB, PLR, PST, REL, REM, REP, RST, SKP, SPC, SYN, TCD, and WST, the location counter will be set to that value modulo 2^{15}. In the second pass, any binary information in the core will be emitted to the current binary-output sink before the location counter is changed. Thus the effect of an absolute decimal location punched in the symbol field is similar to the effect of an ORG with the same number punched in the variable field. However, a symbol (other than those defined by SYN pseudo-operations) defined after an absolute decimal location appears in the symbol field will not be relocatable unless an ORG or LOC pseudo-operation intervenes between the absolute decimal location and the symbol.

If the binary format is relocatable then the relocation of information on these cards will not cause a corresponding relocation of symbols defined when relocatability is suspended. That is, symbols defined in the symbol field after an absolute decimal location behave as though they were defined by the pseudo-operation EQU.

The effect of punching absolute decimal locations on the pseudo-operations specifically excepted above ranges from no effect to phase errors which in turn may lead to a useless assembly.

2. Operational Codes

a. Pseudo-operations. The assembler is controlled by a number of pseudo-operations. These pseudo-operations are instructions to the assembler in much the same way that operations (binary bit patterns) are instructions to the 704.

In the following list of pseudo-operations and their functional definitions, the letters A, X, N, T, etc. appearing in their variable fields are not a part of the pseudo-operation and symbolize some legal variable-field expression:

ABS: Absolute binary format

First pass

Nothing.

Second pass

Binary information (if any) present in core is punched (or written on tape) in the current binary format. The binary format will then be switched to SHARE standard absolute binary.

The assembler's nominal binary format is SHARE standard absolute binary.

AST T, N: Adjoin symbol tables

Same as RST except that instead of replacing any symbol table in core, this pseudo-operation combines the two tables into a single table containing all of the symbols of each.

BCD: Hollerith data

First pass

A symbol appearing in the symbol field is given the value of the current contents of the location counter and v spaces are set aside for the contents of the record. The value of v will lie in the range 0 through 9 if that digit is punched in Column 12. If Column 12 is blank or contains any character other than 0 through 9, then v will have the value 10.

If no symbol is given, identification is relative to that word most recently identified by a symbol.

Second pass

The first v six-character Hollerith words (Columns 13 to 72) are stored in BCD in successive locations.

If, in fact, v equals 0 then no words will be stored, and the BCD card will be treated essentially as a remarks card.

BES N: Block ended by symbol

First pass

The location counter is given the value of the location counter plus the value of the expression N. All symbols appearing in N must be predefined. The symbol appearing in the symbol field is then assigned the value of the location counter (i. e., the value L+N). The symbol refers therefore to that location one beyond the end of the reserved block of N locations.

Second pass

Binary information (if any) present in the core is punched (or written on binary tape) and the assembler location counter is increased by the value of N.

BOS T: Binary-output select

First pass

Nothing.

Second pass

Binary information (if any) present in core is punched (or written on binary tape).

If T is zero or blank, the binary output unit is changed to the on-line punch.

For T = 1, 2, ... 10, the binary output is changed to logical tape T. Rewinds and end of file marks on such tapes are the responsibility of the operator. END and TCD pseudo-operations cause single-word records (corresponding to the 9L row of a transfer card) to be written on the currently selected tape. The assembler's nominal binary output unit is the on-line punch.

BSS N: Block started by symbol

First pass

The symbol appearing in the symbol field is given the current value of the location counter. The counter is then set to the value of the location counter plus the value of the expression N, thus reserving N locations.

All symbols appearing in N must be predefined.

Second pass

Binary information (if any) in core is punched (or written on binary tape), and the assembler location counter is set to the value of the location counter plus the value of N.

C/T A, X: Correction/transfer

First pass

If the value of X is one, the location counter is not changed.

If the value of X is zero, or X is blank, the assembler's location counter is given the value of A.

Neither A nor X may contain an undefined symbol at the time the C/T is encountered.

Second pass

If the value of X is one, the instruction HTR A, X converted to binary is inserted as the next word in the binary-card image. The assembler location counter is not changed. If the value of X is zero, or X is blank, then the instruction HTR A, X is inserted as the next word in the binary-card image and the assembler location counter is given the value of A.

Remarks

The mnemonic code C/T does not print. The printed octal equivalence consists of an address and a tag.

It is not ordinarily sensible to use this pseudo-operation unless the binary format (FUL) for 24 words per card is in effect. This pseudo-operation is used to assist in the assembly of correction/transfer cards (no check sum). It is the coder's responsibility to see that the C/T falls in the left half of the binary-card image.

For a description of the format and function of correction/transfer cards see SHARE Reference Manual 03. 1-04 or the description of UA CSB 1 or PK CSB 2.

An example of the use of this pseudo-operation is given at the end of the description.

DEC: Decimal data

First pass

A symbol appearing in the symbol field is given the value of the current contents of the location counter and c + 1 spaces are set aside for the contents of the record where c is the number of commas in the variable field. If no symbol is given, identification is relative to that word most recently identified by a symbol.

Second pass

The decimal data beginning in Column 12 is converted to to binary and assigned to successive locations. Successive words of data are separated by commas, and the first blank (or character other than 0 to 9, +(12), -(11), point, comma, B, or E) to the right of Column 12 indicates that all punching to the right is a comment. A blank after a comma will cause the formation of a zero as the last word taken from the card. Signs are indicated by + or - (12 or 11 punch) preceding the number, the exponent, or the binary-scale factor. However it is not necessary to use the + sign.

The symbol appearing in the symbol field identifies the first decimal data word on the card. Successive words are identified relative to the first word. If none of the characters Point (.), E, or B appears 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 Point (.) or both appears 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 that follows immediately 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, 1.2345E1, 1234.5E-2, and 12345E-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. The binary-scale factor used in this conversion is the number that follows immediately after the character B. It is 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 assumed to be at the right-hand end. The decimal exponent used in this conversion is the number that follows immediately after the character E. The order of B and E is not significant. For example, 12.345B4, + 1.2345E1B4, and 12345B4E-3 are all equivalent representations of the same fixed-point quantity.

DEF M, N: Define

First pass

Nothing.

Second pass

The N undefined symbols that follow the DEF card are given the values M, M+1, M+2, ... M+N-1. If the N (or, N) are left blank, this process will continue until the undefined symbols following the DEF card are exhausted, another DEF card is encountered, or the symbol table is filled. A DEF card affects a symbol only the first time it is encountered as an undefined symbol.

An undefined symbol that occurs before any DEF card will remain undefined if encountered again after a DEF card. The M and N may be expressions but all symbols must be defined at the time they are encountered in the second pass otherwise the DEF operation will not be executed.

Note that since symbols defined by DEF are defined during the second pass, they will not appear in any symbol table punched by PST or WST. A symbol table containing such symbols may be obtained by the use of FST.

END A: End of assembly

An END record marks the literal end of an individual assembly and causes:

(a) the symbol table to be sorted

(b) the symbol table to be punched if PST was encountered

(c) the symbol table to be searched for multiply defined symbols

(d) the operation table to be sorted if OPD was encountered

(e) the second pass to be called.

Second pass

Any residual binary information in the core is punched followed by a transfer record with the value of the expression A as the transfer address (see TCD, second pass).
The analysis section is then called.

EQU: Equals

First pass

See SYN.

Second pass

See SYN.

Remarks

This pseudo-operation is used when the symbol appearing in the symbol field represents a nonrelocatable program parameter that is not a function of the location of the program in memory.

FIN: Finish

First pass

When this pseudo-operation is encountered, it is treated as if an input end-of-file condition had been encountered. It signals the end of a series of assemblies and either causes a first-pass HPR-type stop if the assembler itself was loaded on-line, or a "push load button" if the assembler was loaded from a tape or drum.

Second pass

This pseudo-operation should never be encountered during the second pass.

Remarks

This pseudo-operation is primarily an operator's control to be used by a 704 operator when he feels the need to avoid end-of-file marks on an input tape.

FUL: Full binary format

First pass

Nothing.

Second pass

Whatever binary information is presently stored in the core is punched (or written on binary tape) in the current binary format and then the format is changed to punch 24 words per card. A transfer card punched by TCD or END will be absolute or relocatable depending upon whether ABS or REL was in force before the FUL control was encountered.

This pseudo-operation will itself print only if sense switch 2 is down.

FST: Final symbol table

First pass

Nothing.

Second pass

A signal is set which is acted on in the analysis section. This pseudo-operation is printed only if sense switch 2 is down.

Analysis section

A symbol table is punched which contains all of the defined symbols used in the assembly including those defined by the pseudo-operation DEF.

HED: Heading

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

Second, two or more of the individual programs may use the same symbols for references that should be unique.

In order to restore uniqueness, the symbols in each program must be changed 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 operating is given. A new heading pseudo-operation will replace the prefix character. Thus several programs having nonunique symbols may be combined by giving the heading pseudo-operation with a unique character before each program.

However, it is sometimes necessary to make cross-references between the individual programs. In order to accomplish this, such references must be written in the following way. Let H be a 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 the heading feature is to be used, all symbols employed 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.

Any Hollerith character may be used as a heading. However, it is recommended that the variable field operators (+, -, *, /, comma, $ and blank) be avoided because it is always difficult to refer to a symbol in a region so headed from a region with a different heading.

In the event that a variable field operator is used the following procedure will set up the necessary references:

```
+          HED
SYM        CLA SOME
AOOSYM     SYN SYM
A          HED
           TRA SYM
           END.
```

It is not possible to say + $SYM, since this implies 0 + 0 $SYM.

A five-character symbol OMMON which is headed C will be confused with the six-character symbol COMMON.

LIB T: Library search

First pass

The library routine identified by the six-character string in the symbol field is obtained from the library tape on unit T and inserted into the assembly on the collated tape. If the PLR switch is off (i. e., no-print condition) a "LIB" record will precede and succeed the LIB routine. The routine identification string (on the LIB card) will not be inserted in the symbol table, but any symbols appearing in the library routine itself are entered and properly defined. The identification string must be absolutely identical to that punched in the symbol fields of the TBL and LIB cards of the desired routine on the library tape.

If T is blank or one, the assembler will look for the routine on tape unit 1 (the standard SHARE library tape unit); otherwise it will look on the tape indicated by the value of T. Routines should be called from a library tape in the order that they are written for most efficient searching. In order to conserve time, all needed routines should be called from a given tape before going to another tape.

If a routine cannot be found, the library tape is rewound and re-searched. If the routine still cannot be found the LIB card involved will be printed during the first pass.

Second pass

A LIB card encountered in the second pass will suspend all but error prints until another LIB card is encountered. Any routine that has one LIB card associated with it on the collated tape necessarily has two. The first is the original LIB card from the input deck and will be printed in lieu of the routine itself. The second is a "phony" to signal the end of the routine.

LOC A: Location specification

First pass

The assembler location counter is assigned the value of the expression A that appears in the variable field. All symbols appearing in A must have been predefined.

Second pass

The assembler location counter is assigned the value of A.

Remarks

The LOC acts just like the ORG except that LOC does not initiate punching. Because of this difference LOC is useful primarily in the assembly of certain types of self-loading cards. Its use leads in some cases to a more useful listing and preserves the integrity of the location counter. An example of its use is given at the end of this description.

OCT: Octal data

First pass

The symbol appearing in the symbol field is given the value of the current contents of the location counter and c + 1 spaces are set aside for the contents of the record where c is the number of commas in the variable field. If no symbol is given, identification is relative to that word most recently identified by a symbol.

Second pass

The octal data in the variable field is converted to binary and stored in successive locations. The octal point is considered to be to the right of the octal word, with the binary point to the right of the 704 word.

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. If the last octal number is followed immediately by a comma, the blank after the comma will be interpreted as a zero word and will be included by the assembler.

If there are 12 octal digits in an octal number, the following equivalences exist with respect to the high-order digit:

-0 = 4 -1 = 5 -2 = 6 -3 = 7.

OPD: Operation definition

First pass

The first three nonblank Hollerith characters in the symbol field are taken as the mnemonic code. The instruction or pseudo-operation that this code is to represent is punched in octal in the variable field. Leading zeros are assumed if the octal number has less than 12 digits. The number may be signed (+ or -). In the case of a 12-digit octal number, the sign may be contained in the lead digit, i. e., 7 is equivalent to -3.

An instruction may not be assigned a tag by this method because the tag bits are abstracted and used to generate the A, T, and D flags where required. A tag of four in the operation implies that an address is normally required. A tag of two implies that a tag is required and a tag of one implies that a decrement is required.

Second pass

The OPD record prints as is. Note that this pseudo-operation can only add a new operation for the current assembly.

ORG A: Origin specification

First pass

The assembler location counter is assigned the value of the expression A appearing in the variable field which must be less than 2^{15}. All symbols occurring in A must have been predefined. If no origin specification is given for the program, the initial value of A shall be zero.

Second pass

Binary information (if any) present in the core is punched (or written on binary tape) and the assembler location counter is assigned the value of A.

PLB N: Push load button

First pass

If N is blank or one a push-load-button sequence is executed at the time the PLB is encountered. The proper return is to READ1 in the first pass.

Second pass

If N is blank or two, the binary information in the core (if any) is punched (or written on tape) and a push-load-button sequence is executed. In addition, the return to the assembler must be to the location CPRT.

This pseudo-op will print only if sense switch 2 is down.

PLR: Print library routine

First pass

A switch is actuated in the LIB-op routine. The switch is normally off. When the switch is off no-print signals (LIB records) are written on the collated tape with each library subroutine. When the switch is on these signals are omitted.

Second pass

This pseudo-operation will itself be printed only if sense switch 2 is down.

PST: Punch symbol table

First pass

If at least one PST control is encountered anywhere before the END card, the symbol table will be punched on-line at the end of the first pass.

Second pass

This control will be printed only if sense switch 2 is down.

REL: Relocatable binary format

First pass

Nothing.

Second pass

Any binary information remaining in the core will be punched or written on tape in the binary format currently in force, and then the binary format will be changed to SHARE relocatable binary.

This control will print only if sense switch 2 is down.

REM: Remark

First pass

Nothing.

Second pass

The contents of Columns 1 to 6 will be printed in the columns occupied by the symbol field of the print out, Columns 7 to 11 will be blank, and Columns 12 to 72 will print in the columns of the variable field.

REP M, N: Repeat

First pass

The values of the expressions M and N are computed and M * N are spaces reserved in addition to the M locations already reserved by the records that are to be repeated. If any symbol in M or N is undefined, the REP record is printed during the first pass and no extra spaces are reserved.

Second pass

The M records preceding the REP operation are repeated N times; thus they appear N + 1 times.

Remarks

Only one word of information may appear on a repeated card. The REP record will be printed if sense switch 2 is down. A symbol defined on a repeated card will print only the first time. The value of the symbol will be the value at that appearance.

RST T, N: Replace symbol table

First pass

A symbol table is read into the core at the time that the RST pseudo-operation is encountered. This replaces the existing symbol table (if any).

For T = 0 or blank, the table will be read from the on-line card reader. A blank card or card reader end-of-file skip signals the end of the symbol table.

For T = 1, 2, ... 10, the logical tape T is rewound, N-1 records having a zero first word are skipped over, and the Nth table is read in to the core. The next record having a zero first word signals the end of the symbol table. Tape T is then rewound.

SKP: Printer channel skip

First pass

Nothing.

Second pass

The character which appears in Column 12 replaces the first character of the next line printed. Any character permissible for printer carriage control (PROGRAM) may be used.

This control will, itself, print only if sense switch 2 is down. If switch 2 is down, this will be the line carrying the special initial character. This operation controls the arrangement of the assembly listing only.

Remarks

The effect of such a first character on the on-line printer is shown in the following table:

		Share board 1 or 2
Character	Standard	UAC modification
(12 punch)	Suppress space	Suppress space
0	Double space	Double space
1 or 2 J to K	Skip to channel 1 or 2	Skip to channel 1 or 2
3 to 9 L to R	Skip to channel 1	Skip to channels 3 to 9

SPC: Paper spacing

First pass

Nothing.

Second pass

If the value of the expression in the variable field is two, then during the assembly output listing, a printer under PROGRAM control will double space until another SPC pseudo-operation is encountered. If the value of the expression is anything other than two or is undefined or if the first character of the variable field is a blank, a printer under PROGRAM control will single space until another SPC is encountered. If no SPC-operation is encountered the printed output will single space.

This operation will itself print only if sense switch 2 is down.

SYN: Synonym

First pass

The symbol appearing in the symbol field is assigned the integer value of the expression appearing in the variable field. Each symbol appearing in the variable field must have been defined before the SYN pseudo-operation is encountered. If any symbol in the variable field is not defined at this time, the image of the SYN record will be printed immediately, and the symbol in the symbol field will not be defined.

Second pass

The value of the variable field is recomputed and printed to the left of the SYN record. If one or more symbols appearing in the variable field have values different from those they had in the first pass (this may be caused by symbols which are multiply defined or defined after the SYN record was first encountered), then the value printed will, in general, differ from the value appearing in the symbol table. If at this time some symbol remains undefined, no value will print (i. e. blanks).

Remarks

This pseudo-operation is used only when the value of the symbol being defined is to be a function of the location of the program in memory (i. e. the symbol is the location of an instruction or piece of data, etc.)

TCD A: Punch transfer card

First pass

Nothing.

Second pass

Any binary information saved in the core is omitted to the selected binary output unit (on-line punch or binary tape) in the selected format. Then a single card is punched with the value of A in the 9L address (i. e. a transfer card) or a single binary word is written on binary tape with the equivalence of A as its address (HTR A), depending on whether the binary output is to be punched or written on tape. A must be previously defined.

If the binary-output mode is ABS, the 9L prefix will be zero. If the mode is REL, the 9L prefix will be two. If the mode is FUL, the value of the 9L prefix will be that of the last binary-output mode other than FUL.

WST T, N: Write symbol table

First pass

The symbol table as it exists at the time the WST pseudo-operation is encountered is sorted. For T = 0 or blank, N (if any) is ignored and the symbol table is punched on-line in SHARE standard absolute binary cards. However, for T = 1, 2, ... 10, logical tape T is rewound and skipped past (N-1) records which have a zero first word. The tape will then be positioned immediately after the (N-1)th symbol table on the tape. The new symbol table (th Nth) is then written in SHARE standard absolute binary card images followed by a single one-word record of zeros and an end-of-file mark. The tape is then rewound.

Note that this procedure requires that for multiple-symbol tables written on a tape, the first should have an N of 1, the second an N of 2, etc.

Second pass

The WST record will be printed if switch 2 is down.

b. Alphabetic codes. In addition to the standard three-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	+0000
PON	Plus one	+1000
PTW	Plus two	+2000
PTH	Plus three	+3000
FOR	Four	-0000
FVE	Five	-1000
SIX	Six	-2000
SVN	Seven	-3000

c. Mnemonic operation codes. A detailed explanation of the Machine Instruction Codes with one exception appear in the IBM-704 Manual of Operation, IBM Form-A22-6500-2. The one exception is the copy-add-and-carry (CAC or CAD) $(-700)_8$ instruction. This instruction is equivalent to a copy and skip instruction (0700) followed by an add and carry logical-word instruction (0631) unless skipping occurs, in which case ACL is omitted. Since the 704 installed at this facility is a drumless machine, all instructions relating to drums are deleted from the summary of machine instructions. In coding symbolic instructions, which have CHS, CLM, COM, DCT, ETM, ETT, IOD, LTM, LBT, PBT, RCD, RPR, RTT, RND, SLF, SPT, SSM, SSP, WPR, or WPU as their operational part, the address part should be blank or zero, since the assembly program automatically introduces the correct address.

In coding symbolic instructions which have BST, RTB, RTD, REW, SLN, SLT, SPR, SPU, SWT, WEF, WTB, WTD, or WTS as their operational part, the address part should be the unit number (in decimal). For instance, BST2 implies back-space tape No. 2, SPR9 implies sense printer exit No. 9, and so on. The assembly program automatically computes the correct octal address (222 and 371 respectively, in the foregoing examples).

(1) Summary of machine instructions

Alphabetical Code	Description	Octal code
ACL	Add and carry logical word	0631
ADD	Add	0400
ADM	Add magnitude	0401
ALS	Accumulator left shift	0767
ANA	And to accumulator	-0320
ANS	And to storage	0320
ARS	Accumulator right shift	0771
BST	Backspace tape	0764
CAC	Copy, add, and carry	-0700
CAD	Copy, add, and carry	-0700
CAL	Clear and add logical word	-0500
CAS	Compare accumulator with storage	0340
CHS	Change sign	0760, 002
CLA	Clear and add	0500
CLM	Clear magnitude	0760, 000
CLS	Clear and subtract	0502
COM	Complement magnitude	0760, 006
CPY	Copy or skip	0700
DCT	Divide check test	0760, 012
DVH	Divide or halt	0220
DVP	Divide or proceed	0221
EFM	Enter floating-trap mode	-0760, 002
ETM	Enter trapping mode	0760, 007
ETT	End of tape test	0760, 011
FAD	Floating add	0300
FDH	Floating divide or halt	0240

FDP	Floating divide or proceed	0241
FMP	Floating multiply	0260
FSB	Floating subtract	0302
HPR	Halt and proceed	0420
HTR	Halt and transfer	0000
LBT	Low-order bit test	0760, 001
LDQ	Load MQ	0560
LFM	Leave floating-trap mode	-0760, 004
LGL	Logical left	-0763
LLS	Long left shift	0763
LRS	Long right shift	0765
LTM	Leave trapping mode	-0760, 007
LXA	Load index from address	0534
LXD	Load index from decrement	-0534
MPR	Multiply and round	-0200
MPY	Multiply	0200
MSE	Minus sense	-0760
NOP	No operation	0761
ORA	Or to accumulator	-0501
ORS	Or to storage	-0602
PAX	Place address in index	0734
PBT	P bit test	-0760, 001
PDX	Place decrement in index	-0734
PSE	Plus sense	0760
PXD	Place index in decrement	-0754
RDS	Read select	0762
REW	Rewind	0772
RND	Round	0760, 010
RQL	Rotate MQ left	-0773
RTT	Redundancy tape test	-0760, 012
SBM	Subtract magnitude	-0400
SLQ	Store left-half MQ	-0620
SLW	Store logical word	0602
SSM	Set sign minus	-0760, 003
SSP	Set sign plus	0760, 003

STA	Store address	0621
STD	Store decrement	0622
STO	Store	0601
STP	Store prefix	0630
STQ	Store MQ	-0600
STZ	Store zero	0600
SUB	Subtract	0402
SXD	Store index in decrement	-0634
TIX	Transfer on index	2000
TLQ	Transfer on low MQ	0040
TMI	Transfer on minus	-0120
TNO	Transfer on no overflow	-0140
TNX	Transfer on no index	-2000
TNZ	Transfer on no zero	-0100
TOV	Transfer on overflow	0140
TPL	Transfer on plus	0120
TQO	Transfer on MQ overflow	0161
TQP	Transfer on MQ plus	0162
TRA	Transfer	0020
TSX	Transfer and set index	0074
TTR	Trap transfer	0021
TXH	Transfer on index high	3000
TXI	Transfer with index incremented	1000
TXL	Transfer on index low or equal	-3000
TZE	Transfer on zero	0100
UFA	Unnormalized floating add	-0300
UFM	Unnormalized floating multiply	-0260
UFS	Unnormalized floating subtract	-0302
WEF	Write end of file	0770
WRS	Write select	0766

(2) Extended Operations

Read		
RCD	Read card reader	0762, 321
RPR	Read printer	0762, 361
RTB	Read tape - binary	0762, 221-232
RTD	Read tape - decimal	0762, 201-212
Write		
WPR	Write printer	0766, 361
WPU	Write punch	0766, 341
WTB	Write tape - binary	0766, 221-232
WTD	Write tape - decimal	0766, 201-212
WTS	Write tapes - simultaneously	0766, 321-325
Sense		
SLF	Sense lights off	0760, 140
SLN	Sense light on	0760, 141-144
SLT	Sense light test	-0760, 141-144
SPR	Sense printer	0760, 361-372
SPT	Sense printer test	0760, 360
SPU	Sense punch	0760, 341-342
SWT	Sense switch test	0760, 161-166
Other		
IOD	Input-output delay	0766, 333

3. Assembly Diagnosis

As an aid to the programmer and installation, the assembly program gives some indications of programming and machine errors:

a. First pass. Certain pseudo-operations require that symbols appearing in their variable fields be defined at the time the operation is encountered in the first pass. These are BES, BSS, C/T, EQU, LIB, LOC, ORG, REP, RST, SYN, and WST. In addition LIB requires that the identification appearing in the symbol field be identical with that of some library routine on the specified tape.

If the assembler is unable to find an equivalence in the proper place, the record in error will print on-line and, if switch 5 is up off-line, with a U to its left. In addition, if any record read from the input tape or any library tape causes four redundancy checks, the record that so failed will cause a similar print but with an R to the left. If the symbol table or operation table is filled in the first pass, an appropriate comment will print on-line and the computer will stop.

b. Second pass.

(1) Off-line print

(a) Certain instructions are tested to see that they have the normally required parts (i. e. address, tag, decrement). If one or more is missing, an appropriate letter prints to the left of the printed instruction in the flag field as shown below:

			Symbolic instruction		
Flags	Core location	Octal instruction	Location	OP	Variable field
ATD	xxxxx	x xxxxx x xxxxx	ALPHA	TXI	
TD				TXI,	
D				TXI, ,	
				TXI, , 0	
D				TXI ALPHA, ,	

(b) If a symbol appearing in the symbol field or variable field of an instruction or pseudo-operation is defined more than once even though the definition is unique, an M will print in the flag field. The particular equivalence used by the assembler is unpredictable but will be the same for a given assembly.

(c) If a symbol in the variable field or the operation code is undefined, a U will print in the flag field.

(d) If the value of a symbol in a symbol field is not the same as it was in the first pass, a P (phase) will print in the flag field and the location counter will be changed to the first-pass value unless the symbol involved is multiply defined.

(e) If an STO follows immediately after a FDH, FDP, DVH or DVP, a Q (quotient) will print to the left of the STO instruction.

(f) If a redundancy check occurs four times, an R will print in the flag field, and a print will be forced on-line whether sense switch 3 is down or not.

(g) In the case of library routines that are not to be printed, failures of the sort described in items 2 through 6 will cause a print in spite of the no-print bias.

(h) The following information about the input tape, the collated tape, and all library tapes will print:

(1) the total number of records read

(2) the number of single failures

(3) the number of double failures

(4) the number of triple failures

(5) the number of quadruple failures.

Note that in the case of the library tape, these will be totals of all library tapes used in that assembly.

(i) The number of input records read on-line will print.

(j) The number of off-line print records generated will print.

(k) The number of defined symbols, symbols defined by DEF, and undefined symbols will print.

(2) On-line print

(a) All symbols defined by DEF will be listed.

(b) All undefined symbols will be listed.

(c) All multiply defined symbols will be listed with their various equivalents.

(d) If the symbol table was full in the second pass, an appropriate comment will print.

4. The Symbol Table

Each symbol that appears in the symbol field of an instruction or is defined by use of a pseudo-operation will be entered in the symbol table as a BCD word with its corresponding core address. Therefore, each symbol requires two cells in the assembler's symbol table area. Unassigned symbols are also recorded in the symbol table with their corresponding core address set to zero. The symbol table may be read-out and used for later assemblies by the use of the WST or PST pseudo-operation codes. These operations will either write the table on a tape or punch it in binary cards on-line. The read-in is accomplished by the operation RST. It should be noted that symbols defined by DEF will not appear in any symbol table recorded by the operations PST or WST. A symbol table containing such symbols may be obtained by the use of the operation FST.

If the symbol COMMON is not defined and reference is made to it, then the corresponding core location for COMMON in the symbol table will be the address of the next cell after the last one assigned in the current assembly. If COMMON has been defined, then it will be recorded in the normal manner.

The maximum number of symbols that may be constructed in the symbol table is 15433 entries. In cases where the program to be assembled uses the library tape, however, the maximum number of symbols that the assembler can handle is somewhat reduced. This follows from the fact that the entire ordered list of subroutines that forms the first 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 15433 minus the number of library subroutines on the tape. The number of library routines will vary from time to time; therefore, if this number is required it should be determined at the time of need.

a. Symbol-table format. All punched-symbol tables begin at $(1900)_{10}$, $(3554)_8$. When the PST pseudo-operation is used (or WST without designating the tape number, the symbol table is punched in binary cards on-line in the following format:

9 row left: Bits 9L13, 9L15, 9L16 punched. Bits 9L24-9L35: loading address

9 row right: Card check sum

8 row left (first card only): 8-left decrement contains 2N, where N is the number of symbols in the table. 8-left address contains a number 2E, which is defined such that $2^{E-1} < 2 * N \leqslant 2^{E}$.

8 row right (first card only): Logical check sum of the entire symbol table (not including the word represented by the 8-left word).

8 to 12 rows left: BCD symbols. The individual symbol is packed to the right, with leading, imbedded, and trailing blanks being replaced by leading zeros. If the symbol has less than six characters (leading zeros not counted), the current heading is inserted at the extreme left end of the word. If, for example, the heading is H, the symbols A, AA, AAA, AAAA, AAAAA, and AAAAAA will lead to entries in the symbol table of HOOOOA, HOOOAA, HOOAAA, HOAAAA, HAAAAA, and AAAAAA, respectively.

The symbols are placed in algebraically increasing sequence.

8-to 12-rows right: In address field, integer equivalences corresponding to BCD symbol in rows left. If the word is negative, the symbol is multiply defined. If the tag is two, the symbol is relocatable; if zero, the symbol is not relocatable.

b. Reassembly features. Additions to a program which has been assembled are easily accomplished if the table of symbols which was written on tape or punched during the initial assembly process has been saved. By use of the AST or RST pseudo-operations, the symbol table may be appended or replaced. It is then necessary only to reload this table and assemble the new parts of the program. The original program need not be loaded. 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 that are to be changed.

5. Standard Boards (On-line)

a. Card reader, type 711. The card reader reads Columns 1 to 72 directly into calculate entry left and right.

b. Card punch type 721. Normally, calculator exits 1 to 72 punch into card Columns 1 to 72. On cycles containing Sense Punch 1, (0760, 341), calculator exits 2 to 9 punch into card Columns 73 to 80. Thereafter data punched into Columns 73 to 80 of this card will be gang-punched in cards following.

c. Printer, type 716.

(1) Board No. 1

(a) "72-72": Calculator exits 1 to 72 into type wheels 1 to 72 in a single cycle.

(b) "72-120": Calculator exits 1 to 72 into type wheels 1 to 72 on first cycle of double cycle; calculator exits 1 to 48 into type wheels 73 to 120 on second cycle of double cycle.

(c) "72-spread-72": Calculator exits 1 to 12 into type wheels 1 to 12 and exits 13 to 72 into an arbitrary combination of type wheels 13 to 120 on a single cycle.

(2) Board No. 2

(a) "72 to 72" with echo-checking possible

(b) "72 to 120" with echo-checking possible

(3) In addition to the above modes of operation, both boards will execute the following functions:

(a) normal single spacing

(b) double-space under control of a sense instruction

(c) normal overflow

(d) overflow suppression under control of a sense instruction

(e) programmable overflow under control of a sense instruction (unsuppressable)

(f) extra space under the control of a sense instruction

(g) space suppression under control of a sense instruction

(h) carriage skip to Channel 2 under control of a sense instruction

(i) a panel check under the control of a sense-exit and the sense-entry instruction.

(4) Board No. 1 uses all 20 co-selectors and eight of the pilot selectors. Board No. 2 uses all 20 co-selectors and 4 of the pilot selectors.

(5) Sense-exit functions

If no senses are given, the page will space one line before each line is printed. (A space means that the paper will be moved one line (1/6 in.) and does not refer to blank lines between printed lines.) Automatic overflow will cause sheet ejection to take place at the end of each page. If 72 to 120 mode printing is called for, this will take place on the same line as the previous 72 to 72 line.

There may be at least 5 msec of computing between the last copy of the copy loop and the sense instructions following the copy loop. The sense instructions given before the copy loop may be given anywhere between the WPR or RPR and the first copy. The order of the sense instructions is unimportant except as they are given before or after the copy loop.

Note also that spacing before printing is suppressed on the first line of a page after an automatic sheet overflow. This is not the case if the page is restored manually or by means of the Sense Printer 1 (SPR1) or Sense Printer 2 (SPR2) instructions.

A sense may be energized either before or after the copy loop for either read printer or write printer. The function of each according to the time at which it is given follows. Note that in general one copy is not equivalent to a full 24 or 46 copies.

Sense printer 1. This is wired directly to skip to Channel 1. If this instruction is given before the copy loop, the page will be ejected before the line is printed; if it is given after the copy loop, the line will be printed and then the sheet will be ejected. The operation of this sense is unaffected by any other senses, either before or after the copy loop.

Sense printer 2. This is wired directly to skip to Channel 2. The operation is exactly like SPR1 except that only a half-page skip occurs, either before or after printing.

Sense printer 3. This sense must be given after the copy loop and causes one space to be made after the line is printed, regardless of the number of spaces called for before printing the line. Note that an automatic overflow may occur after printing the line.

Sense printer 4. This sense must be given before the copy loop, and causes two spaces to be made before printing the line. Note that if these force an automatic overflow, the page will be ejected after the line has been printed. This sense is rendered inoperative if either a SPR5 or SPR9 is also given.

Sense printer 5. This sense must be given before the copy loop, and prevents any spacing before the line is printed. If SPR1 or SPR2 is also given before the copy loop, the page will eject before the line is printed.

Sense printer 6. (board No. 1 only). Sets up format control for the 72-spread-72 mode of operation.

Sense printer 7. Panel check exit.

Sense printer 8. This sense must be given before the copy loop, and prevents automatic overflow before the line is printed, regardless of the number of spaces being made before the line is printed. If one or more spaces are called for after printing, automatic overflow may occur after the line has been printed.

Sense printer 9. This sense must be given before the copy loop, and causes Columns 1 to 48 of the card image to be printed from type wheels 73 to 120. Columns 49 through 72 are printed from type wheels 49 to 72. In addition, spacing is suppressed before the line is printed. If sheet ejection is called for, however, the page will be ejected before the line is printed.

Sense printer 10. Not used.

6. Conventions and Restrictions

a. Machine space. There are no restrictions placed on the use of space in the core or on tapes at compute time except for the built-in restrictions concerning loading and trapping. The programmer does not control storage at assembly time.

b. Symbol "COMMON". The symbol "COMMON" has been established by convention to designate erasable storage for SHARE subprograms. Since it consists of six characters, it may not be headed, and it should not be otherwise restricted. Many library subroutines require blocks of erasable storage to be assembled with them. The initial (symbolic) location of the block to be shared by all programs in the same assembly is "COMMON."

c. Sense switches. Whenever a sense switch is used for control of a program, the "sense switch down" position shall be the unusual case. Sense switch No. 6 will be used for trapping-mode control in those programs relying upon switch setting in such control.

d. Relocatable subprograms. All SHARE programs in relocatable cards will have a standard origin of 0 (zero), with COMMON storage having an origin of $(2000)_8$.

e. Subroutines. The point transferred to shall always be the first instruction in the subroutine.

Subroutines in the library are entered at their initial locations, t, with the following calling sequence:

LOC	OP	ADR	TAG	DCR
z	TSX	t	4	
z + 1	R_1	P_1	R_2	P_2
Z + 2	R_3	P_3	R_4	P_4
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
z + K	Error return			
z + K + 1	Normal return			

The arguments and answers will be placed in the following units of the machine in the order indicated:

(1) Accumulator

(2) M Q

(3) Core-storage location specified in linkage. The P_i in the calling sequence represent parameters, such as core-storage locations or scale factors (negative scale factors should be complements of 2's). The R_i in the calling sequence represent any information that can be conveyed in three bits.

The SHARE subroutines depend on index register No. 4 for their calling sequence. Subroutines written for the library should use this register for its calling sequence. Index registers and sense lights, when used by the subroutine, shall be restored to their original condition within the subroutine before exiting. The status of overflow indicators, the divide check light, the accumulator, and MQ are not guaranteed on exit from a subroutine. Subroutines will not normally use sense switches.

Stops within subroutines should be avoided. Instead exits should be made to the error return with a code placed in the accumulator defining the reason for failure. Explanations of the code will be explained in the program write-up.

f. Program classification. Programs are assigned a two-character classification code. The leftmost character is a letter indicating a primary class; the second character is a digit indicating a secondary class within the primary. The classifications are as follows:

A. Programmed arithmetic
   1. Real
   2. Complex
   3. Decimal

B. Elementary functions
   1. Trigonometric
   2. Hyperbolic
   3. Exponential and logarithmic
   4. Roots and powers

C. Polynomials and special functions
   1. Evaluation of polynomials
   2. Roots of polynomials
   3. Evaluation of special functions
   4. Simultaneous nonlinear algebraic equations
   5. Simultaneous transcendental equations

D. Operations on functions and solutions of differential equations
    1. Numerical integration
    2. Numerical solutions of ordinary differential equations
    3. Numerical solutions of partial differential equations
    4. Numerical differentiation
E. Interpolation and approximations
    1. Table look-up and interpolation
    2. Curve fitting
    3. Smoothing
F. Operations on matrices, vectors, and simultaneous linear equations
    1. Matrix operations
    2. Eigenvalues and eigenvectors
    3. Determinants
    4. Simultaneous linear equations
G. Statistical analysis and probability
    1. Data reduction
    2. Correlation and regression analysis
    3. Sequential analysis
    4. Analysis of variance
    5. Random-number generators
H. Operations research and linear programming
I. Input
    1. Binary
    2. Octal
    3. Decimal
    4. BCD
    9. Composite
J. Output
    1. Binary
    2. Octal
    3. Decimal
    4. BCD
    5. Analog
    9. Composite

K. Internal information transfer
    1. Read-write drum
    2. Relocation
L. Executive routines
    1. Assembly
    2. Compiling
    3. Automatic operator programs
M. Information processing
    1. Sorting
    2. Conversion
    3. Collating and merging
N. Debugging routines
    1. Tracing and trapping
    2. Dumps
    3. Search
    4. Breakpoint print
O. Simulation programs
P. Diagnostic programs
Q. Service programs
    1. Clear-reset programs
    2. Check sum programs
    3. Restore, rewind, tape mark, and load button programs
Z. All others.

C. Operating Procedures - SHARE Assembler

1. Tape Assignment

Logical tape	Function
1.	Share assembler (SAP followed by library programs).
2.	Off-line print.
3.	Off-line input.
6.	Collated tape.

Tapes one and 6 are always required. All tapes must be rewound at the start of assembly.

2. Sense-Switch Settings

Switch 1 up	Off-line input. Tape 3 required.
Switch 1 down	On-line input. Tape 3 not required.
Switch 2 up	Normal print (off-or on-line).
Switch 2 down	Certain pseudo-ops do not normally print. With this switch down, they will print.
Switch 3 up	On-line printer prints only certain error and statistical information.
Switch 3 down	On-line printer prints everything.
Switch 4	Not used.
Switch 5 up	Off-line output tape prepared. Tape 2 required.
Switch 5 down	No off-line output tape required. Tape 2 not used.
Switch 6	Not used.

3. Usage

The Share assembler is the first file on the tape; the library of subroutines is the second file. The Share assembler tape should always be mounted in logical tape unit 1 without the file-protect ring. When all necessary tapes have been mounted and sense-switch settings made, one of two loading procedures is followed:

a. On-line input. Ready symbolic cards in the card reader following an Assembler Tape call card (UA SAP RT).

Press clear button.

Press load button.

For multiple assemblies, each deck of symbolic cards must have an END card as its last card. The decks are then stacked one after the other in the card reader. The last stacked deck may be followed by a FIN card. The function of the FIN pseudo-operation and an end-of-file are similar except that the FIN-op will eject the paper and be printed before the load button is pushed.

b. Off-line input. Ready card reader with assembler-tape call card (UA SAP RT).

Press clear button.

Press load button.

For multiple assemblies, each deck of symbolic cards must have an END card as its last card. The decks are then stacked one after the other and read onto tape via off-line equipment. The last stacked deck may be followed by a FIN card. The presence or absence of a FIN card is treated in the same manner as when input is on-line.

SAP SRCH

It is often convenient to be able to start the assembler operating on some assembly other than the first or to by-pass the remainder of an assembly which caused a stop in the first pass (i. e. for symbol table or operation table full, etc.). A SAP SRCH card will read any tape 3 and cause the following stops:

00007	HTR	End-of-file
00010	HTR	False end-of-record
00017	HTR	END card encountered
00022	HTR	FIN card encountered.

This program is self-loading. Pressing start after a stop causes it to continue. The program does not rewind the tapes, nor does it write an end-of-file on the output tape (2) at any time.

4. Output

When printing the output tape, the carriage should be selected to PROGRAM. Channels 1 and 9 cause an eject.

5. Stops

The location given to the left of an HPR is the actual location of the HPR instruction. (All locations given in octal.)

(00015)	HPR	Drum-read error. Press start to try again. Manual transfer to address of HPR ignores stop. The drum involved is given by the address of location (00016).
(00016)	HPR	Tape-read error. See above. Tape unit involved is given by address part of location (00017).
(00017)	HTR	Error leading built-in binary loader or loader is out of order. Start over.
(00032)	R-W check	End-of-file encountered by built-in loader. Check deck make-up and start over.
(00053)	HTR	Absolute binary-card check-sum error. Press start to read next card. (Not recommended).
(00107)	HTR	Relocatable card encountered by built-in binary loader. Investigate, fix, and start over.
(01060)	HPR(52525)	False end-of-record skip while reading tape in BCD mode. Press start to try again.
(01173)	HTR	Wrong board in printer. Replace with SHARE board No. 1 or No. 2 and press start (present IBM diagnostic board 4P01D will not stop, since it also uses SPR 7 to impulse the SE hub).
(01311)	HTR	End-of-file detected on collated tape before END card. Machine error. Press start to try to go on.

(01412)	HPR	Symbol table full. (Stop follows print-out of comment on-line.) Press start to go on. Results not useful in general.
(01475)	HPR	Operation table full. (Follows print-out on-line.) Press start to go on.
(01571)	HTR	Error while reading library table of contents. Recommend new library tape and restart. Press start to try again.
(01636)	HTR	False end-of-file on collated tape (third pass). Press start to go on. Statistics probably meaningless.
(01663)	HTR	False end-of-file on library tape. Press start to continue. Or library tape incorrectly prepared.
(01710)	HTR	False end-of-file on collated tape after library error. Press start to try to continue.
(02056)	HPR	A FIN card or end-of-file has been detected (only occurs if binary deck loaded on-line). Press start to "push load button."
(02327)	HTR	Symbol-table check-sum error. Press start to go on. (Not recommended.)
(02611)	HTR	Non-Hollerith character detected in Columns 1-72 of a symbolic card read on-line. Correct the card (third from end in stacker), reload the card reader and press start.
(03015)	HTR	False end-of-record skip while reading symbol table from tape or cards. Press start to try to go on.

(03043)	HPR X	Check-sum error on last card or record (tape) read while reading binary symbol table. If reading on-line, press start to go on. If from tape, press start to try again, manual transfer to X to continue.
(03062)	HTR	Symbol-table check-sum error immediately after execution of AST or RST. Press start to continue. (Not recommended.)
(06262)	HTR	Absolute-binary check-sum error on last card read. Occurs while writing assembler on drum or tape.
(06317)	HTR	Relocatable card found while writing assembler on drum or tape.

A SAMPLE SAP CODE.
THIS ROUTINE WILL EVALUATE A BIVARIATE POLYNOMIAL.

Loc.	Pre.	Dec.	Tag	Addr.			
				04000		ORG 2048	
04000	-0	53400	2	04004		LXD P6, J+1	INITIALIZE INDEX REGISTERS.
04001	-0	63400	4	04020	P4	SXD P2, K	STORE K.
04002	0	50000	1	04022		CLA A+1, J	OBTAIN FIRST ELEMENT.
04003	1	77777	1	04004		TXI P6, J, -1	
04004	-2	00001	4	04017	P6	TNX P5, K, 1	
04005	0	76500	0	00043	P3	LRS 35	FORM POLYNOMIAL
04006	0	26000	0	04046		FMP X	IN X
04007	0	30000	1	04022		FAD A+1, J	
04010	1	77777	1	04011		TXI *+1, J, -1	STEP COEFFICIENT
04011	2	00001	4	04005		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
04014	0	26000	0	04047		FMP Y	IN Y
04015	0	30000	0	04051		FAD S	
04016	-3	77754	1			TXL OUT, J, -R/2+1	
04017	0	60100	0	04050	P5	STO Z	
04020	1	00000	4	04001	P2	TXI P4, K	
				00005	N	EQU 5	
				00052	R	EQU N*N+3*N+2	Note that the a_{ij} are stored in the
				04021	A	BSS R/2	order $a_{05}, a_{14}, a_{04}, a_{23}, a_{03}, \cdots$
04046	0	00000	0	00000	X		a_{00} from location A on.
04047	0	00000	0	00000	Y		
04050	0	00000	0	00000	Z		
04051	0	00000	0	00000	S		
				00001	J	EQU 1	
				00004	K	EQU 4	
				04000		END P4-1	
				00001	0	OUT	

SHARE ASSEMBLER STATISTICS

TAPE	TOTAL	1 FAIL	2 FAIL	3 FAIL	4 FAIL
INP	0	0	0	0	0
LIB	0	0	0	0	0
COL	32	0	0	0	0

NUMBER OF ON-LINE INPUT RECORDS 32

NUMBER OF OFF-LINE PRINT RECORDS 42

NUMBER OF SYMBOLS, DEF 14, DEFOP 0, UNDEF 1

A USE OF THE CONTROL OPERATION LOC.

A SELF-LOADING REMOTE BOOTSTRAP FOR A 1 TO 17 INSTRUCTION PROGRAM. THE BOOTSTRAP USES LOCATIONS 0, 1, R, AND R+1. THE PROGRAM WILL OCCUPY LOCATIONS R+2 TO R+N+1, WHERE N IS THE NUMBER OF WORDS IN THE PROGRAM. AFTER THE PROGRAM IS LOADED, THE FIRST INSTRUCTION EXECUTED WILL BE THE ONE IN LOCATION R+2.

```
    Loc.     Pre.    Dec.    Tag  Addr.
                                  00000        ORG 0
                                  01000   R    SYN 512
    00000    0       70000   0    01000        CPY R
TD  00001    1       00000   0    01000        TXI R
                                  01000        LOC R
    01000    0       70000   0    01001        CPY R1
    01001    1       00021   0    00000   R1   TXI 0,,N
                                  01000        LOC R
    01000    -0      53400   4    01001        LXD R1,4
                                  01000        LOC R
    01000    0       70000   4    01023        CPY R+N+2,4
    01001    2       00001   4    01000        TIX R,4,1
                                     CONTINUE HERE WITH THE PROGRAM WHICH IS TO BE
                                     LOADED.  CARD SIZE LIMITS THE PROGRAM TO 17
                                     INSTRUCTIONS.
                                  00021   N    EQU 17      PARAMETER=THE NUMBER OF PROGRAM WORDS.
                                  00000        END **
```

SHARE ASSEMBLER STATISTICS

TAPE	TOTAL	1 FAIL	2 FAIL	3 FAIL	4 FAIL
INP	0	0	0	0	0
LIB	0	0	0	0	0
COL	27	0	0	0	0

NUMBER OF ON-LINE INPUT RECORDS 27

NUMBER OF OFF-LINE RECORDS 35

NUMBER OF SYMBOLS, DEF 3, DEFOP 0, UNDEF 0

A USE OF THE CONTROL OPERATION C/T

A CORRECTION/TRANSFER CARD TO PUSH THE LOAD BUTTON WHEN LOADED BY UA CSB 1 OR THE EQUIVALENT.

```
      Loc.     Pre.   Dec.    Tag  Addr.
                                    00000          ORG 0
                                    00005    T     SYN 5
                                 1  00005               T, 1    TRANSFER ADDRESS          9L ROW
A     00000    0      00000    0  00000          HTR            DUMMY FOR CARD SPAC-      9R ROW
                                    00005               T       ING.                      8L ROW
      00005    0      76200    0  00321          RCD            FIRST LOCATION            8R ROW
                                 1  00000                 , 1   READ CARD READER          7L ROW
      00006    0      70000    0  00000          CPY 0          LOCATION COUNTER +1       7R ROW
                                 1  00000               , 1     ETC.                      6L ROW
      00007    0      70000    0  00001          CPY 1                                    6R ROW
                                 1  00000               , 1                               5L ROW
      00010    0      02000    0  00000          TRA 0                                    5R ROW
                                    00000          END -
```

SHARE ASSEMBLER STATISTICS

TAPE	TOTAL	1 FAIL	2 FAIL	3 FAIL	4 FAIL
INP	0	0	0	0	0
LIB	0	0	0	0	0
COL	19	0	0	0	0

NUMBER OF ON-LINE INPUT RECORDS 19

NUMBER OF OFF-LINE PRINT RECORDS 27

NUMBER OF SYMBOLS, DEF 1, DEFOP 0, UNDEF 0

III. THE FORTRAN-II SYSTEM

A. Operational Procedures for the Fortran II Compiler

1. Tape Assignment

a. Nonbatch compile:

(1) Set the system tape to tape 1.

(2) Set off-line input tape to tape 2 (for source program on tape prepared off-line).

(3) Ready tapes 1, 2, 3, and 4.

Note: Tape 2 is always readied regardless of whether input is on cards or tape.

b. Batch compile:

(1) Set the system tape to tape 1.

(2) Set off-line input tape to tape 5 (for source programs on tape prepared off-line).

(3) Ready tapes 1, 2, 3, 4, 5, 6, and 7.

Note: Tape 7 is not required if every source program calls for punched-card output. Tape 5 is always readied regardless of whether input is on cards or tape.

2. Sense-Switch Settings and Tape Output

a. Sense switch 1.

Up: Binary cards for the object program(s) are punched on line. Tape 3 contains the binary program for the last or only program compiled. Tape 7 is not used.

Down: Binary cards for the object program(s) are not punched. Tape 3 contains the binary program for the last or only program compiled. For batch compiling, tape 7 contains the binary programs for all the source programs compiled in the order they were compiled.

b. Sense switch 2.

Up: Produces, on tape 2, two files for the source program compiled, which contain the source program statements and a map of the object program storage. For batch compiling, tape 6 will contain two files for each program compiled and tape 2 will contain two files for the last program compiled.

Down: Adds a third file for each program compiled (see above) containing the object program in SAP-type language on tape 2 (and tape 6 for batch compiling).

c. Sense switch 3.

Up: No on-line listings are produced.

Down: Lists on-line the first two or three files on tape 2, depending on the settings of sense switch 2.

d. Sense switch 4.

Up: Causes Fortran II to produce a program optimized with respect to index registers.

Down: Causes Fortran II to produce a program not fully optimized with respect to index registers but which will be translated more rapidly.

e. Sense switch 5.

Up: Library routines are not to be punched out or written on tape 3.

Down: Causes Library routines to be punched on-line or written on tape 3, depending on whether sense switch 1 is in the up or down position.

f. Sense switch 6.

Up: Single program compilation.

Down: Batch compilation.

3. Usage

a. Printer: SHARE printer board No. 2.

b. Card Reader:

(1) On-line input; Ready source program cards in card reader.

(2) Off-line input; Ready card reader.

Note: Compiler attempts to read source programs from card reader and goes to tape for programs if end-of-file condition is encountered from card reader. Tape 2 is always required, regardless of input mode.

c. Depress load-tape button.

d. Compiling. Successful compiling is indicated as follows:

(1) Single problem: Card reader is selected. Depress READY button. Program stops location 35_8.

(2) Batch compile: Card reader is selected. Depress READY button. Program stops at location 35_8.

4. Punched-Card Output

If sense switch 1 is UP, Fortran II produces the following punched-card output:

a. Compilation of a main program:

BSS loader (9 cards)

Program card (9 punch in Column 1)

Program in relocatable binary (9 punch in Column 2)

Transfer card (9 punch in Column 1, remaining card blank).

b. Compilation of a subprogram:

Program card (9 punch in Column I)

Program in relocatable binary (9 punch in Column 2).

Note: Columns 1 to 36, row 7 of program card for subprogram contains BCD representation of the name assigned.

Columns 1 to 36, row 7 of program card for main program are blank.

See reference manual for Fortran II, C28-6000-1, for further description of object cards.

5. Executive System Halts

Section number	Rec. No.	Octal loc	Stop source	Procedure and details
1-CS	000	27	Tape 1	Press START to try reading Tape 1 (system tape) again. Tape 1 has been read once unsuccessfully because of either Redundancy error or check sum error in reading in System record.
1	013	30	Machine error	If not batch compiling, press START to rerun problem. For batch compiling, press START to rerun current problem; or, turn on sense light 1 and press START if next problem is to be compiled.
PRE 1	001	0147	Program	Remove cards from card reader and run out the cards in the reader. There is an impossible character (non-Hollerith) in the third card from the last in the stacker. Correct the invalid character before recompiling.
Successful compilation record	009	35	Job completed	Compilation is complete. All source programs have been compiled, or an attempt at compilation has been abandoned because of source-program or machine error. Computer control is returned to the installation via a load-button sequence. If the card reader is ready, but empty of cards, this halt results.

Source program error record	010	50	Source cards or tape 5	Compilation is complete. There has been a source-program error if the program is in a single-problem compile mode. This halt can also result if the END card in a batch compilation is missing or mispunched, or if tape 5 cannot be read successfully. Computer control is returned to the installation via a load-button sequence. If card reader is ready, but empty of cards, this halts results.

B. Executing the Object Program

1. Usage

a. Binary and subroutine card outputs. The binary card output of a main program compilation consists of the following sequence of cards:

(1) BSS loader (9 cards)
(2) Program card
(3) Program in relocatable binary
(4) Transfer card (9 punch in Col. 1).

The binary card output of a subroutine or function subprogram compilation consists of:

(1) Program card
(2) Program in relocatable binary.

Where a permanent or general library subroutine is called by a program compilation that does not suppress punching of subroutine cards, the subroutine card output will be:

(1) Program card
(2) Program in relocatable binary.

b. Loading. In order to run the object program, the Fortran II object program deck must consist of the following sequence:

(1) BSS loader
(2) Program card
(3) Program in relocatable binary
(4) Program card
(5) Program in relocatable binary
(6) Program card
(7) Program in relocatable binary

(n) Transfer card (9 punch in Col. 1).

One of the programs must be a main program and all the others subprograms. The programs may be in any order.

The deck is readied in the card reader and the LOAD CARD button is pressed.

(1) All the subprograms called for in the main program and other subprograms must be in the deck if it is to run. Of course, they normally will be there as a result of a compilation with sense switch 5 down. If any are missing, a stop at 77756 or 77775 will occur during loading.

(2) Although duplicate subroutines taken from the library tape will never occur in a single main program, subroutine, or function compilation, they may easily occur in a main program - subprogram sequence. If this occurs, the duplicate copies will be loaded, although only the first of these will ever be called during execution.

(a) To save core space during execution, the duplicate subroutines should be extracted from the compiled deck. This may be accomplished by searching for the program card that identifies the program and removing it along with its binary cards.

(i) The program card is identified by a 9 punch in Column 1 and by punching in the 8 row words. The transfer card, which also has a 9 punch in Column 1, has no punches elsewhere. The program card of a subprogram is distinguished from the program card of a main program by not being blank in the 7 row.

(ii) The physical sequence of subprograms belonging to any main-program, subroutine, or function compilation is the exact reverse of their appearance in that section of the object program storage map labelled "Subroutines Punched from Library."

(b) In order to save both compilation time and card-searching time caused by duplications of library subroutines, binary-card copies of certain of these frequently used subroutines may be kept aside and inserted into the load deck when needed. This would enable some programs to be run with sense switch 5 up, which otherwise could not be.

(3) The transfer card must be the last card in the load deck. It is, however, compiled as the last card in the main program, and the main program may not be the last program in the deck ready for loading. In this case, two alternatives are available:

(a) The transfer card may simply be extracted and placed at the end of the complete load deck.

(b) Another transfer card (9 punch in Col. 1) may be placed at the end of the load deck. In this case a stop at 77775 will occur during loading at the time the first transfer card is encountered. Pressing the START button enables loading to continue.

(c) The control card is used to relocate lower memory locations upwards in cores, and common data downwards in cores. That which is referred to in the Fortran II Manual as the "Common Reassignment Card" is a control card with reference only to the relocation of COMMON. When the control card is used, it must be placed immediately before the program card of the program concerned.

2. Error Halts in the BSS Loader

Halt (octal)	Reason for halt	Procedure
3	Instructions and symbol table of loader overlap.	Get off machine. Combination of program and transfer vectors too long. Rewrite program.
20	End of file in the card reader.	Press START to read more cards.
77453	Instructions and data overlap.	Get off machine. Combination of instructions and data too long. Rewrite program.
77556	Check sum error.	Press START to accept information.
77756	More than 20 subroutines are missing.	If missing subroutines are at hand, press START until stop at 77775_8 is reached. Follow instructions (a) for that stop.
77775	Missing subroutines.	This stop indicates the transfer card has been reached. It is caused by one of two things: (a) Loading has been completed, but at least one of the subroutines called for is missing. Location 77453 contains the BCD name of the first missing subroutines, location 77454, the second, etc. If the missing subroutine(s) is immediately available, it may be loaded without starting the entire loading process over again. Place another transfer card (9 punch in Col. 1) at the end of the routine(s), ready the card reader, and press START.

(b) The transfer card encountered is really a premature one that simply has not been withdrawn. Be certain that a transfer card is the last card at the end of the deck, and press START.

3. Error Halts in the Object Program

There are 11 standard error halts in the object-program level input-output routines. They are to be recognized not by looking at the instruction counter but by looking at the HPR instruction itself in the storage register:

Halt	Reason for Halt	Procedure
HPR 0, 0	End of file in reading binary tape.	Press START to begin reading next file.
HPR 0, 1	End of file in reading cards or BCD tape.	Press START to begin reading next file.
HPR 1, 1 HPR 2, 1 HPR 3, 1 HPR 4, 1	Inappropriate character encountered in a data field in reading cards or BCD tape.	Pressing START causes that character to be treated as a zero.
HPR 5, 1	Illegal control character in FORMAT statement.	Press START to continue.
HPR 0, 2	Non-Hollerith character encountered in reading input cards.	Correct card, ready the card reader and press START. Do not press START before correcting card(s).

HPR 0, 3	Redundancy check in reading BCD tape.	Press START to accept information read.
HPR 0, 4	Echo check in printing	Press START to continue. Press RESET and then START to repeat line, and then continue.
HPR 7, 7	Binary-tape error in reading binary tape.	Press START to accept information read.

Note: The error halt HPR 7, 7 in octal, as it would appear in the storage register, is shown as 042000700007.

C. Library Subroutines

SQRTF(X)

Computes the square root of |x|. If x is negative, $\sqrt{+x}$ will be computed. There is no error stop.

LOGF(X)

Computes natural log of |x|. If x is negative, ln(+x) will be computed. There is no error stop.

EXPF(X)

Computes e^x for $|x| < 87.3$. For $x \geq 87.3$, e^x is set equal to 10^{38}. For $x \leq -87.3$, e^x is set equal to ϕ.

SINF(X)

Computes sin(x) for $2^{-8} < (x) < 2^{29}$. For $x \geq 2^{29}$, sinx is ϕ. For $x < 2^{-8}$, sinx is X.

COSF(X)

Computes cos(x) by adding $\pi/2$ to x and computing sine as above.

ATANF(X)

Computes arctangent of any floating-point number x(in radians).

TANHF(X)

Computes hyperbolic tangent of x for $|x| < 5896644 \times 10^{30}$.
For $|x| > 11.0903540$, tanh x = 1.
For $|x| > 5896644 \times 10^{30}$, tanh x is set to 128 and the accumulator-overflow light is turned on. This routine turns off the divide-check light.