RECOMP II USER'S PROGRAM NO. 1118

PROGRAM TITLE:	AFAR SYMBOLIC ASSEMBLER MANUAL
PROGRAM CLASSIFICATION:	Executive and Control
AUTHOR:	BROADVIEW RESEARCH CORPORATION
PURPOSE:	To translate into machine language the symbolic coding generated by AFCOR (An Algebraic Compiler).
DATE:	May 1961

Published by

RECOMP User's Library

at

AUTONETICS INDUSTRIAL PRODUCTS
A DIVISION OF NORTH AMERICAN AVIATION, INC.
3400 E. 70th Street, Long Beach 5, Calif.

FOREWORD

This manual describes the operation and use of the symbolic assembler prepared under contract AF 23(601)-2857 for the Aeronautical Chart and Information Center, U.S. Air Force, by Broadview Research Corporation.

The operation and use of the algebraic compiler, which can be used to produce symbolic assembler-language programs, is described in BRC 161-10, AFCOR Algebraic Compiler Manual.

The flow charts and coding for the compiler and assembler appear in the following documents:

BRC 161-11-I, AFAR Symbolic Assembler Flow Charts

BRC 161-11-II, AFAR Symbolic Assembler Coding

BRC 161-14-I, AFCOR Algebraic Compiler Flow Charts

BRC 161-14-II, AFCOR Algebraic Compiler Coding

TABLE OF CONTENTS

Section 1
INTRODUCTION AND GENERAL DESCRIPTION

The Assembly Routine is a two pass program which translates coding written for the Recomp II computer from a symbolic language to an object program in explicit binary form required by the computer. In describing the use of this program it is assumed that the user is familiar with the Recomp and its operation as explained in Autonetics publications.*

In the following sections, the symbolic language is described in full, including the exact rules for coding, permissible and non-permissible formats, and all details of the use of the assembly routine.

Eighty-column punched cards are assumed as the principal input medium. The most common sequence is assumed to be:

1. Write the program on coding paper.
2. Punch cards -- one card per line of coding.
3. Convert from cards to Baudot paper tape with a Systematics converter.
4. Assemble -- load the assembly program tape and the input tape prepared in 3. above. Output is a punched tape object program and an optional typewriter listin listing.

On-line typewriter input may replace steps 2. and 3. above.

---

* 1. Operating Manual for Recomp II (Revised May, 1959)
2. Supplement to Recomp II Operating Manual (April, 1960)
3. Recomp II Technical Bulletin No. 11 (June 3, 1960)

The alternate sequence for generating an object program is:

1. Write the program in compiler language.
2. Punch cards.
3. Convert from cards to Baudot paper tape with Systematics converter.
4. Compile.
5. Assemble.

Section 2
CODING FORMAT

Each coding line consists of six fixed-length fields. These fields are symbolic location, operation, L or V loop designation, symbolic address, absolute address, and numeric, and their use is described below.

SYMBOLIC LOCATION (SL)

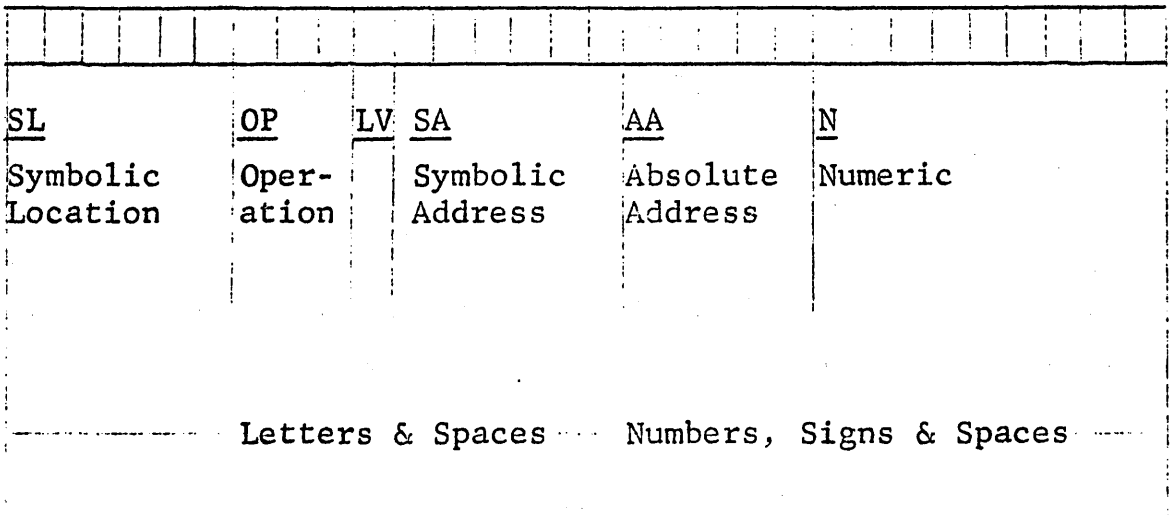
This field is six characters long. It is the first field of the coding line (Fig. 1) and must contain letters or blanks only. The normal purpose in using a location symbol is to give a name to the instruction with which the location symbol is associated, so that the instruction may be referred to by this name in other instructions of the program.

The symbol -- if any -- appearing in this field must begin in the first column of the field and must contain no imbedded blanks (i.e., be packed left). Symbols identifying machine instructions must begin with A through H.* Symbols identifying fixed point quantities must begin with I through N.* Symbols identifying floating point quantities must begin with O through Z.*

Every symbol appearing in a program must be defined by appearing in SL or by use of a SYN pseudo-operation (see Section 3). If it does not appear in one of the above ways, the symbol is said to be undefined. If it appears more than once, it is said to be multiply defined and, of course, is ambiguous as a name.

---

* These conventions correspond to half, full, and double computer word usages.



Examples:

SL	OP	L V	SA	AA	N
ABCDEF	CLA		JA	+2973	
	STO	L	ABSDEF		
JA	DEC			+1.7436	
JX	OCT			+1764352116453	
XW	BSS	XW		20	

Note: All fields must be packed left.

Figure 1. Coding Line

Although there is nothing logically wrong in naming a location without ever using that name, it is generally desirable to use a location symbol only if this symbol is referred to elsewhere in the program. The reason is that the assembly program, in processing the source program, keeps in memory a symbol table of location symbols which is limited to 512 entries.

In processing SL, the assembly program uses a location counter. This is an absolute binary number (13 bits) denoting the memory location which the instruction (with the given location symbol) will occupy when the object program is loaded.

OPERATION (OP)

This field is three characters long (see Figure 1) and must contain letters only (blanks are not permitted). Every permissible operation code belongs to one of two classes: it is either a Recomp II machine operation (such as "CLA", "SQR", etc.) or it is a pseudo-operation.

Machine operations always generate 20 bits of absolute machine language in the object program. Pseudo-operations, unlike machine operations, may generate more than 20 bits (i.e., 40, 80 or more) in the object program, or none at all.

Permissible machine and pseudo-operations are described in Section 4. All non-permissible OP codes* are replaced by HTR in the object program.

L OR V LOOP DESIGNATION (LV)

The LV field is one character long (see Figure 1). Permissible characters are "L", "V", or blank. This field is used with

---

* A blank OP field is used by the assembly program as a special sentinel.

machine operations only, and modifies addresses so that they refer to the L or V high speed loops.

The assembly program calculates an absolute address (13 bits) for each machine operation. If LV contains an L(V) the nine high-order bits of the absolute address are replaced by 776 (777). For example, 44311 becomes 77611 (77711). A blank LV implies no modification. Any other character gives an error print.

SYMBOLIC ADDRESS (SA)

This field is six characters long (see Figure 1). SA normally contains a symbol defined in an SL field.* The absolute binary equivalent of SA is augmented by the AA and LV fields to form the 13-bit address of this instruction in the object program. If the symbol in SA is undefined (i.e., does not appear in the symbol table), an error message is typed and the 13-bit address is set to zero.

ABSOLUTE ADDRESS (AA)

This field is five characters long (see Figure 1). The first character must be +, -, or blank. Blank and + are equivalent. AA normally contains a signed decimal number.** This number is converted to binary. If the first character of SA is A through H or blank, the number is unchanged. If the first character of SA is I through N (O through Z), the number is multiplied by 2(4). The resulting number is added to the binary equivalent of SA to form a 13-bit address for this instruction. The half word

---

* For other usages, see BSS, HED, PNC, PTC, REM, SYN, and TYC below.

** For other usages, see ALS, ARS, BSS, CMD, DEC, DSC, FLD, FSC, OCT, ORG, PNC, PTC, REM, TYC below.

bit is set to:

0 for machine operations DIS, PNW, PTW, TYW and all other operations in which a half word bit of zero is used to specify a different operation;

or to:

1 for machine operations DSD, PND, PTD, TYD, and all other operations in which a half word bit of one is used to specify a different operation.

NUMERIC (N)

This field contains nine characters (see Figure 1). It is used only for numeric input and REM pseudo-operations. For usage, see CMD, DEC, DSC, FLD, OCT, REM.

Section 3
OPERATION CODES

Permissible OP codes can be classified into two sets:

1. Machine operations explained in detail in Autonetics publications. (See page 1.)
2. Machine operations and pseudo-operations peculiar to this assembly program. A detailed explanation of these OP codes is included in this section.

OP codes not appearing on the permissible OP code list are replaced by OP code HTR (77 octal) and an error message is printed. (See Section 6.)

PERMISSIBLE OP CODES

Alpha Code	Octal Code	Operation
ADD	Ø1 ... Ø	Add
ADM	Ø1 ... 1	Add magnitude
ALS	41	Accumulator left shift
ARS	4Ø	Accumulator right shift
BSS	None	Block started by Symbol
CAM	ØØ ... 1	Clear and add magnitude
CFL	65	Copy from L-loop
CFV	67	Copy from V-loop
CLA	ØØ ... Ø	Clear and add
CLS	Ø2 ... Ø	Clear and subtract
CMD	None	Command input
CSM	Ø2 ... 1	Clear and subtract magnitude
CTL	64	Copy to L-loop
CTV	66	Copy to V-loop
DEC	None	Decimal input
DIS	36 ... Ø	Display
DIV	22 ... Ø	Divide
DKM*	21 ... 1	Divide by magnitude single length and round
DRM	23 ... 1	Divide by magnitude and round
DSC	None	Decimal scaling
DSD	36 ... 1	Display decimal
DSL	2Ø ... Ø	Divide Single Length
DSM	2Ø ... 1	Divide by magnitude single length

* Listed as SRM in Recomp II Technical Bulletin No. 11 (June 3, 1960)

Alpha Code	Octal Code	Operation
DSR	21 ... Ø	Divide single length and round
DVM	22 ... 1	Divide by magnitude
DVR	23 ... Ø	Divide and round
EXT	33	Extract
FAD	Ø4 ... Ø	Floating add
FAM	Ø4 ... 1	Floating add magnitude
FCA	3Ø	Cloating clear and add
FCS	34	Floating clear and subtract
FDM	Ø5 ... 1	Floating divide magnitude
FDV	Ø5 ... Ø	Floating divide
FLD	None	Floating decimal input
FMM	Ø7 ... 1	Floating multiply magnitude
FMP	Ø7 ... Ø	Floating multiply
FNM	45	Floating normalize
FSB	Ø6 ... Ø	Floating subtract
FSC	None	Floating scaling
FSM	Ø6 ... 1	Floating subtract magnitude
FSQ	44	Floating square root
FST	35	Floating store
HAL	None	Halt after loading
HED	None	Head
HTR	77	Halt and transfer
MPR	13	Multiply and round
MPY	11	Multiply
NOP	4ØØØØØØ	No operation
OCT	None	Octal input
ORG	None	Origin
PNA	74776XØ	Punch alpha

Alpha Code	Octal Code	Operation
PNC	74 ... Ø	Punch character
PND	14 ... 1	Punch decimal word
PNW	14 ... Ø	Punch word
PTA	76776XØ	Punch and type alpha
PTC	76 ... Ø	Punch and type character
PTD	16 ... 1	Punch and type decimal
PTW	16 ... Ø	Punch and type word
REM	None	Remark
RDY	71	Read Y
RDZ	73	Read Z
SAL	None	Start after loading
SAX	15	Store A and exchange A and X
SBM	Ø3 ... 1	Subtract magnitude
SLA	42 ... Ø	Store left address
SLL	None	Set location left
SLR	None	Set location right
SLZ	None	Set location zero
SQM	25 ... 1	Square root magnitude
SQR	25 ... Ø	Square root
SRA	42 ... 1	Store right address
STA	42	Store address
STO	6Ø	Store accumulator
SUB	Ø3 ... Ø	Subtract
SYN	None	Synonym
TMI	51	Transfer on minus
TOV	53	Transfer on overflow
TPL	52	Transfer on plus
TRA	57	Transfer

Alpha Code	Octal Code	Operation
TRP	None	Trap next instruction
TSB	54	Transfer on Sense Switch B
TSC	55	Transfer on Sense Switch C
TSD	56	Transfer on Sense Switch D
TYA	72776XØ	Type alpha
TYC	72	Type character
TYD	12 ... 1	Type decimal word
TYW	12 ... Ø	Type word
TZE	5Ø	Transfer on zero
XAR	43	Exchange A and R

For interpretation of OP codes not contained in the following section, see Autonetics publications:

1. Operating Manual for Recomp II and Supplement
2. Technical Bulletin No. 11.

ALS Accumulator Left Shift SL, OP, AA*

The AA field is converted to binary and put into the six bits in the sector address. See the Recomp II Operating Manual for further details.

ARS Accumulator Right Shift SL, OP, AA

See ALS

BSS Block Started by Symbol SL, OP, SA, AA

This operation is used to specify blocks of memory reserved for such purposes as data storage, erasable storage, etc. The AA field contains the number of storage locations to be saved. This number is controlled by the first letter of the symbol in the SA field. If this first letter is A through H, half words (or instruction) storages are saved. If the first letter is I through N, full word storages are saved. If the first letter is O through Z, double word (or floating) storages are saved.

In the case of full word storage (I through N), if the current value of the location counter specifies a right half word, this location is skipped (i.e., set to minus zero) and the block is started in the following full word. In the case of floating storage (O through Z), the location counter is set to the nearest following even full word location, and the skipped locations set to minus zero, before the block is saved.

---

* Fields used by each OP code are indicated following the operation name.

If the SA field is blank, an error message is typed and the block is saved as though floating storage were specified.

CMD ComManD input SL, OP, AA, N

This operation causes the contents of the AA field and the first three numbers in the N field to be put into the half word location at the current value of the location counter. The first character is the sign, the next six characters are octal, and the remaining character is binary.

DEC DECimal SL, OP, AA, N

The contents of the AA and N fields are converted to binary and put into the full word location at the current value of the location counter unless this is a right half word location, in which case this location is skipped (i.e., set to minus zero) and the converted number is put into the following full word. If the AA and N fields contain no decimal point, the number is assumed to be an integer. If there is no sign, the number is assumed to be positive and the sign column must be blank. The assembly program terminates the number when it finds a blank, so imbedded blanks are not permissible. Binary scaling and additional decimal scaling are specified by means of a DSC operation. If there is no DSC operation given, the binary scaling is assumed to be 39 and the decimal scaling is assumed to be zero (see DSC).

DIS DISplay SL, OP, LV, SA, AA

This operation causes output, as described in the Recomp Operating Manual, in command format only (i.e., the half word bit in the address is a Ø). For decimal display see DSD.

DSC Decimal SCaling OP, AA, N

This pseudo-operation generates no words in the object program but is used to specify scaling -- both decimal and binary -- for DEC operation codes. The decimal scaling is a signed integer in the AA field and the binary scaling is a signed integer in the N field. The decimal scaling gives the power of ten by which the number in the DEC input line is to be multiplied after conversion to binary for insertion into the computer. The binary scaling specifies the position of the binary point in the binary word cell which receives the fixed point binary number resulting from the conversion of the decimal number. This integer is used to count from the left hand end of the binary word cell to the right, so that a binary scaling of ∅ says that the binary point is immediately to the left of bit position 1 and a binary scaling of 39 says that the binary point is immediately to the right of bit position 39. This scaling factor can then be thought of as the number of integral places. No binary scaling implies a scaling of 39 (i.e., an integer).

The DSC operation precedes one or more DEC operations and controls the scaling until a line with any operation other than DEC is reached. At this point, the scaling reverts to a binary scaling of 39 and a decimal scaling of zero.

DSD DiSplay Decimal SL, OP, LV, SA, AA

This operation causes output as described in the Recomp Operating Manual under DIS in decimal format only (i.e., the half word bit in the address is a 1). For command format display, see DIS.

FLD FLoating Decimal SL, OP, AA, N

The contents of the AA and N fields are converted to normalized floating binary and put into the two full word locations at the current value of the location counter plus K (K = 0, 1, 2, or 3 half words), where K is a number such that the fractional portion of the floating number begins in an even left location. If K = 1, 2, or 3, the K half word locations skipped are cleared (i.e., set to minus zero). If the AA and N fields contain no decimal point, the number is assumed to be an integer; if the sign column is blank, the number is assumed to be positive. The number is terminated by a blank, so it must contain no imbedded blanks. Decimal scaling is specified by a FSC operation. If no FSC operation is given, the decimal scaling is assumed to be zero (see FSC).

FSC Floating SCaling OP, AA

This pseudo-operation generates no words in the object program. The AA field contains a signed integer which is the power of ten by which the number in the FLD input line is to be multiplied after conversion to binary.

An FSC operation is used to precede one or more FLD operations, and controls the scaling until a line with any operation other than FLD is reached. At this point, the scaling reverts to zero. The FSC - FLD sequence converts numbers modulo 10^{128}.

HAL Halt After Loading OP, SA, AA

The last instruction in a program must be an HAL or SAL pseudo-operation. The assembly terminates on sensing one of these pseudo-operations. This pseudo-operation generates no words in the object program. This operation, along with SAL, has three functions:

first, it signifies the end of the program to be assembled; second, the SA and AA fields define the address to which the location counter is to be set after loading the object program; third, it causes a halt (start) code to be punched at the end of the object program tape.

HED HEaD OP, SA

This pseudo-operation generates no words in the object program. The SA field of the HED pseudo-operation contains a single character. All instructions following the HED are modified by insertion of this single character between the first and second characters of the symbols in the SL and the SA fields, until another HED pseudo-operation is encountered.

If the SA field of the HED pseudo-operation is blank or contains more than one character, following instructions are <u>not</u> modified; thus a HED pseudo-operation with a blank SA field must be used to terminate a heading operation. If a symbol containing six characters is headed, the right-most character will be destroyed.

NOP No OPeration SL, OP

The NOP operation is replaced by an ARS Ø instruction in the object program.

OCT OCTal input SL, OP, AA, N

The thirteen or fewer octal digits (including a sign) which are contained in the AA and N fields are entered into the full word location at the current value of the location counter, unless this is a right half word location, as an octal integer. If the OCT operation occurs at a right half word location, one half word is skipped (i.e., set to minus zero) and the number is put into

the next full word location. If the sign column is blank, the number is assumed to be positive. The number is terminated by a blank so it must contain no imbedded blanks. If a non-octal digit is encountered, an error message is typed and the number is set to zero.

ORG ORiGin OP, AA

No word is generated in the object program by this pseudo-operation, but the effect is to set the location counter. The AA field is converted from decimal to full word binary. Thus "ORG 64" would set the location counter to octal 100.0. If the AA field is negative, or is greater than 4095, or contains a non-numeric character, an error message will be printed and the location counter will not be changed. A blank AA field will set the location to zero.

PNC PuNch Character SL, OP, SA, AA

The SA or AA portion of this instruction contains the actual character to be output as described in the Recomp Operating Manual. If the character is represented on the typewriter in letter shift mode (i.e., A-Z or control function), it must appear in the SA field. If the character is a figure (Ø-9 or punctuation), it must appear in the AA field. The control functions appearing in the SA field are represented by:

CR - Carriage Return
SP - Space
FS - Figure Shift
LF - Line Feed
LS - Letter Shift
TF - Tape Feed

PND PuNch Decimal SL, OP, LV, SA, AA

This operation causes output, as described in the Recomp Operating Manual under PNW, in decimal format only (i.e., the half word bit in the address is a 1). For command format output, see PNW.

PNW PuNch Word SL, OP, LV, SA, AA

This operation causes output, as described in the Recomp Operating Manual, in command format only (i.e., the half word bit in the address is a Ø). For decimal output, see PND.

PTC Punch and Type Character SL, OP, SA, AA

See PNC.

PTD Punch and Type Decimal SL, OP, LV, SA, AA

See PND.

PTW Punch and Type Word SL, OP, LV, SA, AA

See PNW.

REM REMark SL, OP, LV, SA, AA, N

This pseudo-operation generates no words in the object program but merely provides a method of typing remarks on the symbolic output listing, if any. All characters on the line are typed with the typewriter in letter shift so no numeric characters may appear.

SAL Start After Loading OP, SA, AA

See HAL.

SLA Store Left Address SL, OP, LV, SA, AA

This operation is replaced by an STA (42_8) operation code. If the combined SA and AA portion of this instruction refers to a right

address (.1), an error message is printed. The address is not changed.

SLL Set Location Left . OP

This operation has one of two effects on the object program. If the current value of the location counter specifies a left address, no instruction is generated in the object program. If the current value of the location counter specifies a right address, a NOP instruction is put into the object program at this point.

SLR Set Location Right OP

This operation has one of two effects on the object program. If the current value of the location counter specifies a right address, no instruction is generated in the object program. If the current value of the location counter specifies a left address, a NOP instruction is put into the object program at this point.

SLZ Set Location Zero OP

This operation causes the location counter to be advanced to the nearest higher location, the least significant four bits of which are zero. The locations which are skipped are cleared (i.e., set to minus zero) in the object program.

SRA Store Right Address SL, OP, LV, SA, AA

This operation is replaced by an STA (42_8) operation code. If the combined SA and AA portion of this instruction refers to a left address (.0), an error message is printed. The address is not changed.

SYN SYNonym SL, OP, SA, AA

This pseudo-operation generates no operations in the object program, but merely sets the SL field equal to the SA + AA fields.

If neither of these fields has been defined by a previous line of coding, an error message is printed and the symbols are not put into the symbol table. If both symbols have already been assigned, their previous assignments are unchanged and an error message is printed.

TRP TRaP OP

This pseudo-operation generates no operations in the object program, but sets the sign of the following instruction minus in the object program.

TYC TYpe Character SL, OP, SA, AA

See PNC.

TYD TYpe Decimal SL, OP, LV, SA, AA

See PND

TYW TYpe Word SL, OP, LV, SA, AA

See PNW.

Section 4
PREPARATION OF INPUT TAPE

The symbolic alpha input tape for this assembly program may be prepared either from cards or on-line.

The card format is shown in Figure 2. Columns 1 through 30 are to be punched in the format described in Section 2. Columns 31 and 32 and 65-80 must be left blank. Columns 33 through 64 may contain remarks. These remarks are ignored by the card-to-tape converter.

When the program is ready to be put on tape, a beginning-of-record card should be put on the front of the deck. End-of-record cards, followed immediately by beginning-of-record cards must then be inserted in the deck at intervals of no more than 332 cards. An end-of-record card must be put at the back of the deck (following the HAL or SAL card). Put the assembly board into the converter, the proper drum cards on the drums, the program deck into the hopper, and start the card-to-tape converter.

Leaders at the beginning and end of the tape must be punched manually.

The alternate (on-line) method of tape preparation is used as follows:

1. Set the left hand margin of the Recomp typewriter at zero, with tabs at columns 6, 16, 21, and 30.
2. Ready the tape punch (feed out a suitable leader).
3. Load the assembly tape.
4. Press Start 3. The typewriter carriage will return. The computer stops with the typewriter in letter shift.

5. Type one input line. Note that no letter shift should be typed, but that a figure shift must be typed in column 17 if figures follow.* Fields may be skipped by tabbing. If an error occurs anywhere in the line, tab over to column 30 and type "X" or "/". The line will not be punched. If the line is satisfactory, tab (or type) to column 30 and type "LETTER SHIFT". This character signals no error. The computer will then construct and punch this line. At the completion of punching, the typewriter carriage will return. Note: the typewriter is again in letter shift.
6. Repeat step 5 for each input line.
7. When a line containing either HAL or SAL in the OP field is sensed,** the computer will punch the line, then punch C, carriage return, S, then transfer to the first pass of the assembly routine.
8. Feed out the tape. The program typed is now ready for assembly.

Note: This routine counts the input lines, and punches end-of-record and beginning-of-record information as needed.

---

* In the case of the REM pseudo-operation code, do not type figure shift.

** All programs must end with either a HAL or SAL since the assembly terminates on sensing one of these.

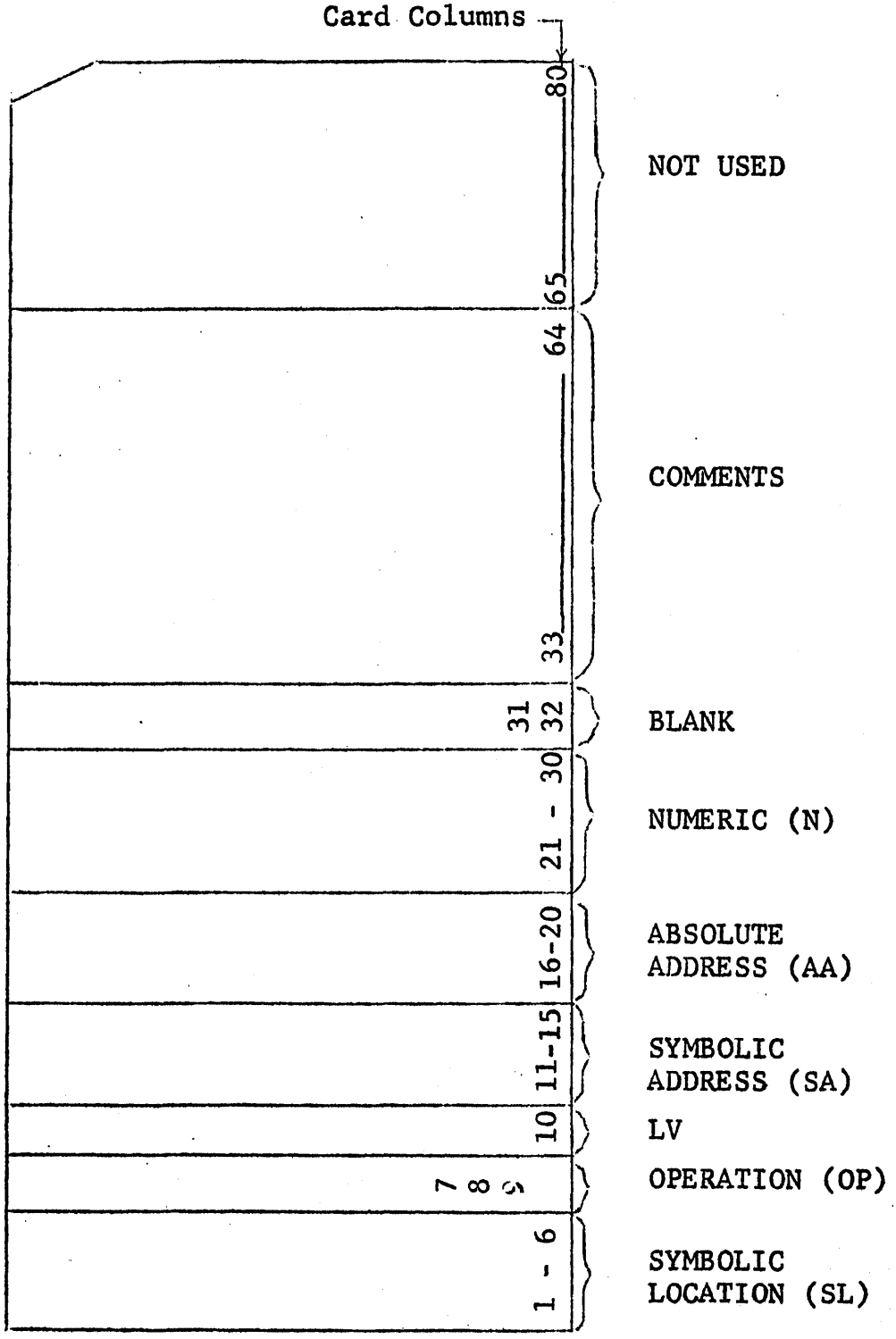


Figure 2. Input Card Format

Section 5
COMPUTER OPERATION AND OUTPUT

This section describes the operating procedures for using the assembly program once the symbolic alpha input tape has been prepared.

Set the typewriter margin at 0, with the first tab at 24. Load the assembly program tape and push Start. The typewriter will type "READY TAPE, PRESS START 1". Ready the input tape in the reader and press the Start 1 button.

Pass One of the assembly reads the input tape, constructs the symbol table, types detected errors, types a memory map showing locations used, and types "END OF PASS ONE, READY TAPE, PRESS START 2", and halts. (Error prints are listed in Section 6.)

If it is desired to complete the assembly, again ready the input tape, and press Start 2. Pass Two of the assembly always punches the object program tape in command format. If Sense Switch B is up, a typewriter listing is also produced, containing a command format print of the object program, an alpha print of the input tape, and detected error prints.

When the assembly finishes processing either of the pseudo-operation codes "HAL" or "SAL", the assembly types "SSC DOWN FOR MULTIPLE ASSEMBLY - PRESS START". If Sense Switch C is left in the up position, the location set and H (or S) code is punched on the object program tape, and the computer stops at location 7777.1_8. If Sense Switch C is set in the down position at this time, the assembly does not punch the location set and H (or S) code on the object program tape but transfers to the beginning of Pass One.

Section 6
ERRORS

Pass One will type detected errors as XXNNNN, where XX is a letter code specifying the error and NNNN is the object program octal location of the word in error. For example, OP2173 signals an error in the OP field at octal location 2173 in the object program.

Pass Two error prints are typed immediately, before the line in which the error occurs, and consist of letter codes only, as explained below.

Codes	Error
A	Inadmissible character in either SA or AA field
AA	Inadmissible character in the AA field
AN	Inadmissible character in the combined AA and N fields
LV	Inadmissible character in the LV field.
MF	Map storage full. If more than 14 ORG pseudo-operation codes appear on the input tape, this error print will occur, and subsequent blocks will not be included in the memory map.
N	Inadmissible character in the N field.
OP	Non-permissible OP code
SA	Error in the SA field
SB(SN)	SB(SN) is typed during processing of an SYN pseudo-operation code if both (neither) symbols have been previously defined.
SC	Improper scaling on a DEC pseudo-operation (overflow)

Codes	Error
SF	Symbol tape full. A maximum of 512 symbols are stored. The computer halts after this print. Pressing "START" will cause the assembly to continue. No further symbols will be entered in the symbol table.
SL	Symbol in the SL field has been previously defined (and is now multiply defined).