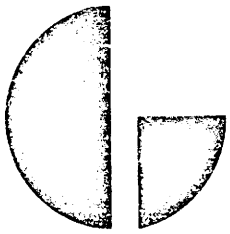


DATA GENERAL
CORPORATION
Southboro,
Massachusetts 01772
(617) 485-9100

PROGRAM

User's Manual

NOVA ASSEMBLER
FOR THE IBM 360, CDC 6600, UNIVAC 1108

ABSTRACT

The Data General Assembler permits assembly of NOVA symbolic source code into machine object code for Data General's NOVA line of computers, using card input to the IBM 360, CDC 6600, or UNIVAC 1108. Output can be in absolute or relocatable assembly mode for Data General's binary or relocatable loaders.

093-000054-00

Data General Corporation is responsible for the maintenance of the NOVA Assembler for the IBM 360, CDC 6600, and UNIVAC 1108, including correction of any bugs found in the program as it is described in this manual. However, users of the Assembler are solely responsible for writing and debugging of any modifications to the program necessary to tailor it to their specific software configurations.

CONTENTS

Chapter 1	-	GENERAL DESCRIPTION	1-1
Chapter 2	-	MACHINE REQUIREMENTS	2-1
Chapter 3	-	INPUT	3-1
		Option Card Format	3-1
		Option Fields	3-1
		File-Assignment Fields	3-3
		Option Card Examples	3-3
		Op-code Deck	3-4
		Source Code Deck	3-5
		Examples of Deck Setups	3-6
		Normal Assembly	3-6
		Assembly with New Op-codes	3-6
		IBM 360-OS	3-7
		CDC 6600	3-7
		UNIVAC 1108	3-8
Chapter 4	-	OUTPUT	4-1
		Listing	4-1
		Source Text Listing	4-1
		Sample Source Text Listing	4-2
		Cross Reference Table	4-3
		Type of Assembly and Error List	4-3
		Sample Cross Reference and Error Listings	4-3
		Object Code	4-4
APPENDIX A	-	Program and Subroutine Description	
APPENDIX B	-	Description of Common Blocks	
APPENDIX C	-	Control Variable Values	
APPENDIX D	-	Character Code Conversions	
APPENDIX E	-	Error Codes	

CHAPTER 1

GENERAL DESCRIPTION

The NOVA/SUPERNOVA relocatable assembler converts symbolic source code into machine object code for the DATA GENERAL NOVA series mini-computers. The expanded features of this assembler make it particularly attractive to the user who has access to any medium or large scale computer system. In addition, his use of an offline assembler with punched card input frees the user's NOVA/SUPERNOVA from the time-consuming tasks of assembling and editing paper tape, thus allowing more efficient use of the mini-computer during program development.

FORTRAN IV was chosen as the language for the assembler due to its almost universal implementation on commercial computer systems. Use of FORTRAN also allows simpler modification of the assembler to better suit individual user requirements.

The assembler output can be formatted for either the binary or the relocatable loader depending on the source code and the user requirements. This object code file can then be copied by a user-written program to a medium suitable for the desired loader.

An additional feature of the assembler is the option card. Through the use of this card the user may specify an absolute or relocatable assembly mode, request the creation of an object code output file, have local symbol blocks added to that file, or specify the file reference numbers to be assigned for assembler I/O. The most significant feature of this card is the "Load Op-codes" option. This allows the user the ability to modify permanent op-codes peculiar to his installation, or add permanent symbols such as device codes. So that he doesn't have to include this option with each assembly, a set of cards is written onto the system punch file which can be incorporated into the assembler as the permanent op-codes.

The user will find that the utilization of this "offline" assembler with all its added features will increase the capabilities of his NOVA/SUPERNOVA installation.

CHAPTER 2

MACHINE REQUIREMENTS

Since the assembler is written in FORTRAN IV, it can be run on virtually any medium or large scale commercial computer system with minimal modification. The only requirements in addition to FORTRAN capability are the input and output files which are described in Chapter 3. Normally, a load medium such as paper tape is required. Due to the lack of standardization in paper tape facilities, the user will have to write a program which will copy the generated object code to the load medium.

The program is supplied on a magnetic tape in 9-track form for the IBM 360/370 and in 7-track forms for the UNIVAC 1108 and the CDC 6600. The tapes may be converted to cards, etc., using standard utility programs. The 9-track tape is in ORG card format and has been created using utility program DEBE. The 7-track tapes are in ORG card format and have been created using an IBM 1401.

Actual job control cards or their equivalent may have to be changed to effect compilation as a function of the machine configuration used.

CHAPTER 3

INPUT

This section describes input to the assembler. Input consists of the option card, a possible op-code deck, and the source code to be assembled.

OPTION CARD FORMAT

The first card read by the assembler is the option card, which specifies assembler options. The option card must be read from file reference No. 5. A blank option card will cause all default values (given under OPTION FIELDS and FILE-ASSIGNMENT FIELDS) to be assumed.

The option card has the format:

option-fields, field-assignment-fields

where: option-fields consists of one or more assembly options.

file-assignment-fields consists of one or more assignments of files to file reference numbers.

The data fields of the option card may be given in any order. Each field is separated from the previous field by a comma. The possible options and files to be assigned are:

{ABASM / RLASM}, LDOPS, LOCLS, NXREF, PTAPE, IN=n, OUT=n, PT=n, SCR=n

where: each n is an integer constant representing the reference number of a file.

ABASM is the default assembly mode option (absolute assembly.)

Option Fields

Assembly Mode: { ABASM / RLASM }

Both absolute and relocatable assemblies are available. If the user wants an absolute assembly of a source code containing no relocation pseudo-op codes, he should specify ABASM. The object code generated can then be read by the binary loader.

If the user wants a relocatable assembly of source code, he should specify RLASM. The object code generated can then be read by the relocatable loader.

Unless otherwise specified, an absolute assembly will be assumed.

Should an absolute assembly be specified when the code contains relocation op-codes, the code will be assembled as relocatable.

Cross Reference Table: NXREF

The cross reference table is described in the output section. If NXREF is not specified, the cross reference table will be output. NXREF causes suppression of the table.

Object File Output: PTAPE

Normally, the user wishes to load the generated object code into his NOVA or SUPERNOVA. To do this, he must specify the creation of an object file which can then be copied to the medium suitable for loading (e.g., paper tape).

No object file will be written if PTAPE is not specified.

Local Symbol Table: LOCLS

If the user wants to utilize the Data General symbolic debugger, he must specify that a local symbol table be included in the object file generated.

If LOCLS is not specified, no local symbol table will be output.

Load New Op-codes: LDOPS

The set of permanent symbols maintained in the assembler can be replaced by the user. To do this, he must prepare a deck of new op-codes. These will replace the op-codes supplied during the execution of the assembler and write a set of DATA cards on the punch file. The DATA cards may then replace those supplied in the BLOCK DATA subprogram. (The BLOCK DATA subprogram supplies initial values to COMMON/SYMTB/.)

LDOPS inactivates <u>all</u> supplied op-codes.

If LDOPS is not specified, the assembler assumes that the deck following the option card is the source deck to be assembled.

Preparation of the op-code deck is described later in this chapter.

FILE-ASSIGNMENT FIELDS

Four files may be specified with the file reference numbers to be assigned to them:

IN	-	Source code file.
OUT	-	File on which the listing will be written.
SCR	-	The scratch file.
PT	-	Object code file.

File assignments are of the form:

```
file=number
```

All files are used sequentially and can be either magnetic tape or direct access.

Default assignments of files are the following:

IN=5, OUT=6, SCR=8, PT=9

Option Card Examples

```
RLASM, IN=7, OUT=6, PT=12, PTAPE, SCR=3
```

The example specifies a relocatable assembly with the source code on file reference No. 7, the listing written on file reference No. 6, the object code written on file No. 12, and with file reference No. 3 as the scratch file. The following example produces identical results:

```
PT=12, RLASM, OUT=6, PTAPE, SCR=3, IN=7
```

Note that in the example, the local symbol table will not be included in the output and that the cross reference table will be included with the listing.

If the option card is blank, default options and file assignments will result. A blank option card is equivalent to:

```
ABASM, IN=5, OUT=6, SCR=8, PT=9
```

OP-CODE DECK

The op-code deck has three fields separated by at least one blank column. The op-code deck is terminated by a blank card. The deck follows the option card when the LDOPS option has been specified.

The fields of the op-code deck are:

Symbol Field: The symbol must begin with a letter or period and may contain up to 5 characters. The characters permitted are alphanumerics and periods.

Value Field: The symbol value is an octal number which is used in processing the symbol.

Type Field: The symbol type is a decimal number that generally determines the way a symbol will be interpreted and processed by the assembler. Symbols added by LDOPS are permanent symbols and, as such, will have 32 added to their type when the op-code deck is read.

Some examples of possible op-code cards are:

Symbol	Value	Type
LDA	020000	14
INTA	061477	10
TTO	11	1

To add a symbol as a device code for a CRT display, one might use:

CRT	31	1

Since the symbol is absolute, the type is 1.

SOURCE CODE DECK

It is assumed that the reader is familiar with the NOVA/SUPERNOVA assembly language as described in Data General publications 093-000017 (Assembler) and 093-000040 (Extended Assembler). The format accepted by this assembler is identical to that described in the manuals with the following qualifications:

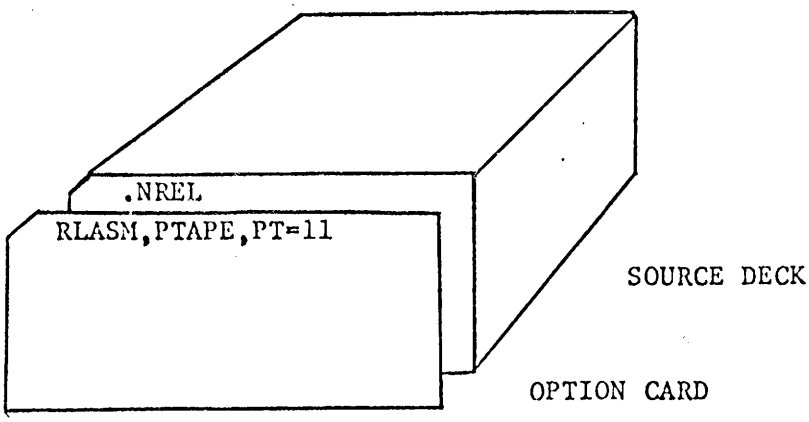
1. The input medium is 80-column punched cards or punch card images on a storage medium.

2. Each punch card contains what is punched on paper tape between two carriage returns.

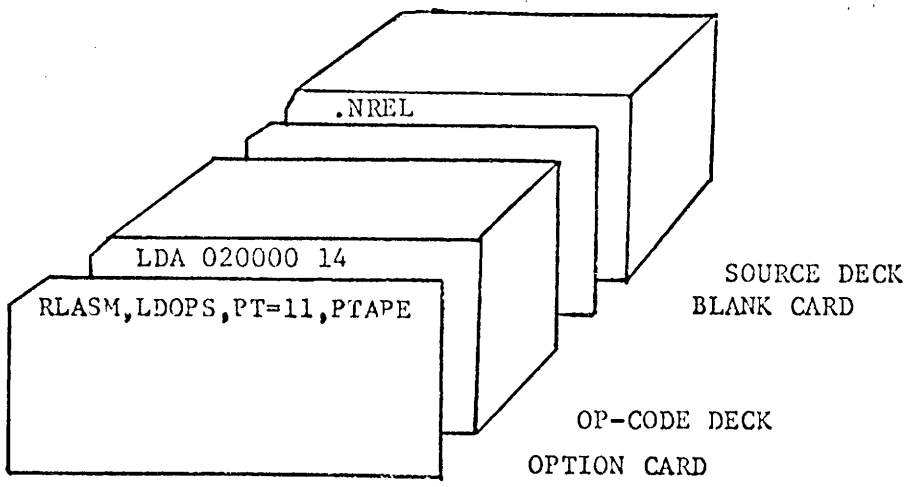
3. The end of tape (. EOT) pseudo-op has no meaning, since the assembly input is one file.

DECK SETUP EXAMPLES

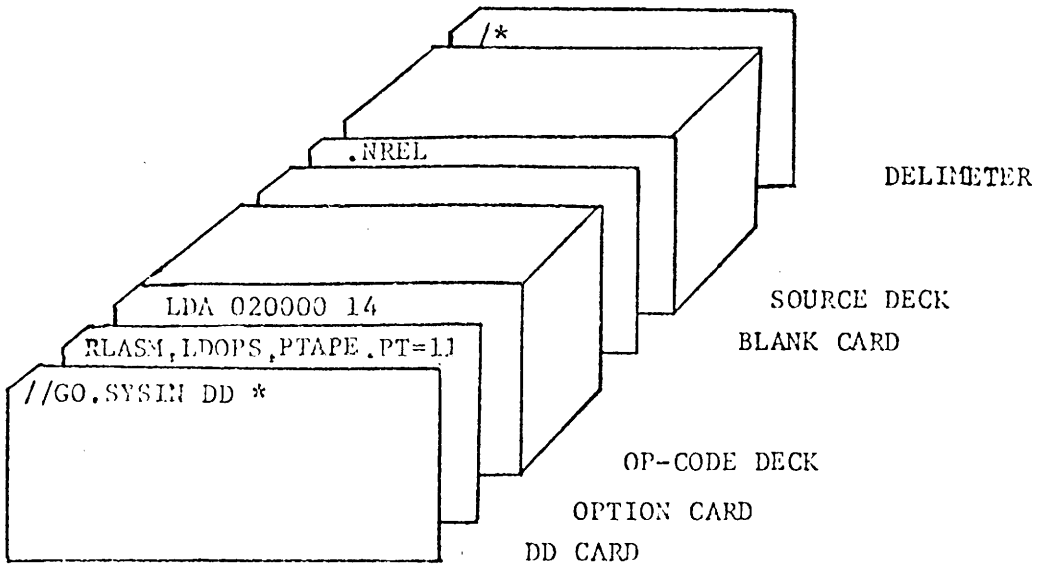
NORMAL ASSEMBLY



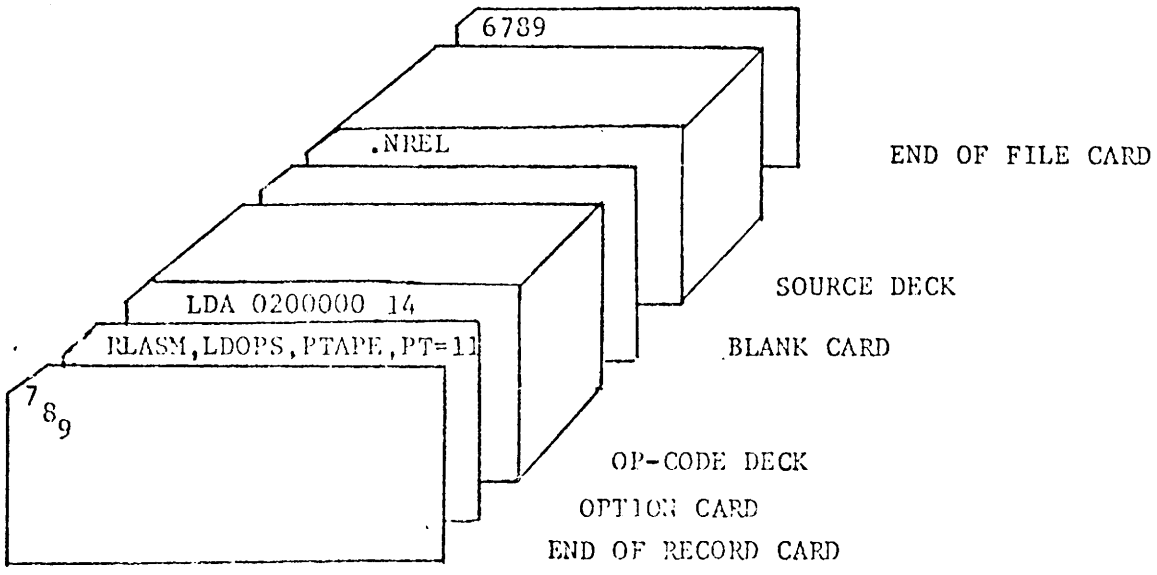
LOAD NEW OP-CODES



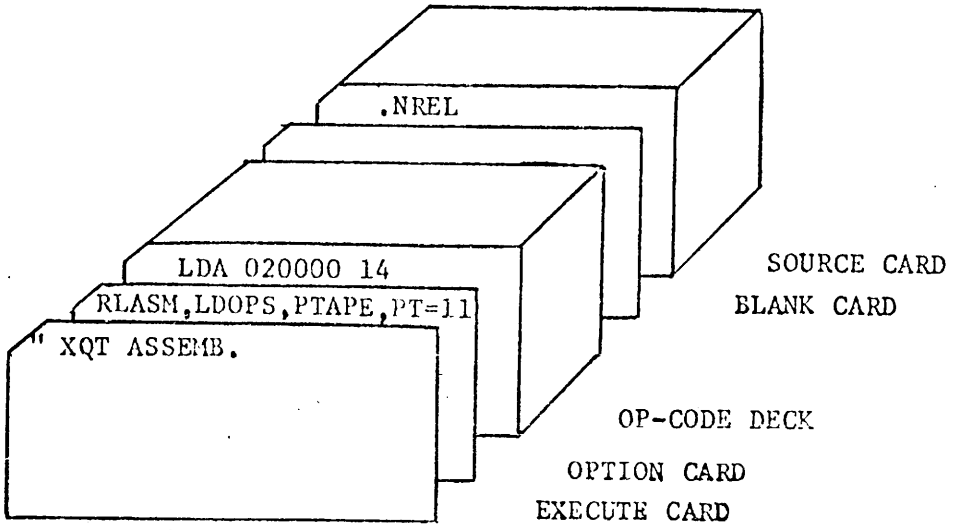
DECK SETUP FOR IBM-360-OS



DECK SETUP FOR CDC-6600



DECK SETUP FOR UNIVAC 1108



CHAPTER 4

OUTPUT

LISTING

Source Text Listing

The listing is created in pass 2 in parallel to the object code output. The format of the listing is similar to that described in Data General publications 093-000017 (Assembler) and 093-000040 (Extended Assembler).

Each source card generates one line in the listing, except for text statements that may generate up to 40 lines.

Each line of the listing has the following fields, moving from left to right across the page:

1. Card ordinal number.

2. Any errors found. The field may contain up to 10 characters, each of which indicates an error found on the card. A list of error codes and their meanings is given in Appendix E.

3. Storage location. The location field gives the octal core location at which the value will be loaded by the loader. If the assembly is relocatable, the relocation symbol is appended; relocation symbols are:

   a. Absolute (blank)

   b. Page zero relocatable (-)

   c. Normally relocatable (')

4. Value. The value field has the octal representation of the 16-bit word which will be loaded at the storage location. If the value is relocatable, a relocation symbol is appended. In addition to the relocation symbols defined above, the following relocation symbols are legal for value:

   a. Page zero byte relocatable (=)

   b. Normally byte relocatable (")

5. Input card. The source input is reproduced in this field.

A sample source listing is reproduced on the following page.

Sample Source Text Listing

```
        NOVA/SUPERNOVA ASSEMBLER VERSION 1.01

 CARD    ERRORS  LOC     VALUE           INPUT CARD
ORDINAL
 NO.
   1               00000   000000          0
   2                       000027          .LOC 27
   3               00027   000170  A:      2*TABL
   4               00030   000113  B:      TABL+17
   5                       000074          .LOC .+43
   6                       000020  TABL:   .BLK 20
   7                       000000-         .ZREL
   8               00000-  003510  PNTR:   SUBRT
   9               00001-  000000' PNTR1:  MAIN
  10                       000007-         .LOC .+5
  11               00007-  000000  ARG1:   0
  12                       000377          .LOC ARG1-PNTR+370
  13               00377   000000  ARG2:   0
  14                       000000'         .NREL
  15               00000'  022027  MAIN:   LDA 0,@A
  16               00001'  024030          LDA 1,B
  17                       000010-         .LOC ARG1+1
  18               00010-  010074          ISZ TABL
  19    L                  000011-         .LOC PNTR1+2
  20               00011-  020006          LDA 0,ARG1-PNTR1
  21                       003510          .LOC 3510
  22               03510   024007- SUBRT:  LDA 1,ARG1
  23               03511   030377          LDA 2,ARG2
  24    L                  003512          .LOC 3500
  25               03512   010400          ISZ SUBRT+2
  26                       000006'         .LOC MAIN+6
  27               00006'  052027          STA 2,@A
  28                       000000          .END
```

Cross Reference Table

Unless an NXREF option is specified, a cross reference table will be produced following the listing. The cross reference table is an alphabetized list of the symbols defined in the preceding assembly. The three fields in the table are:

1. Symbol name.

2. Location assigned to the symbol. If relocatable, the appropriate relocation symbol is included.

3. The card ordinal numbers of cards on which the symbol is referenced.

Type of Assembly and Error List

Below the cross reference table is a line indicating whether the assembly is absolute or relocatable.

The assembler outputs the number of cards on which errors occurred and lists the numbers of the cards that contained errors.

Sample Cross Reference and Error Listings

Following is the cross reference table and error list for the source text listing shown on the previous page.

```
        CROSS REFERENCE TABLE
NAME    LOC & TYPE      CARD ORDINAL
                           NUMBER
A       000027          15   27
ARG1    000007-         12   17   20   22
ARG2    000377          23
B       000030          16
MAIN    000000'          9   26
PNTR    000000-         12
PNTR1   000001-         19   20
SUBRT   003510           8   25
TABL    000074           3    4   18


     * RELOCATABLE ASSEMBLY *


0 THERE WERE AT LEAST   2 ERRORS IN THIS ASSEMBLY

ERRORS OCCURRED ON CARDS    19   24
```

OBJECT CODE

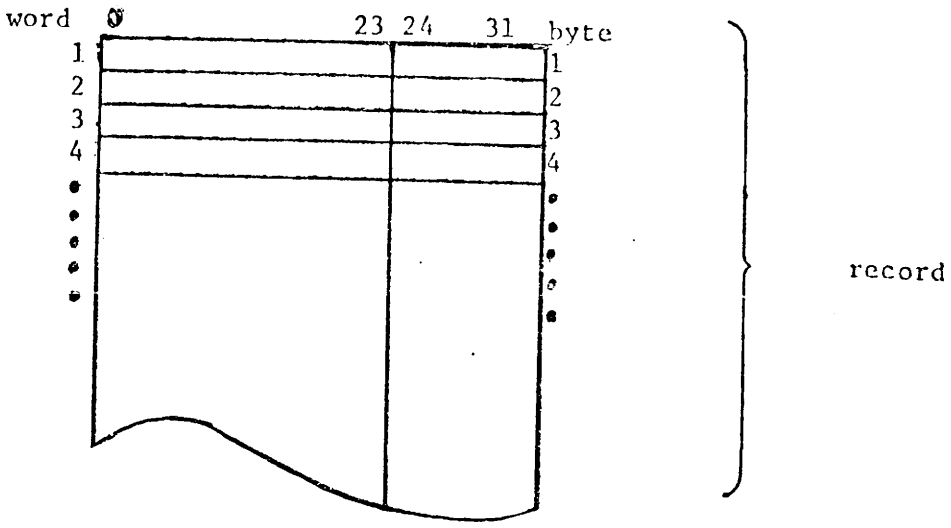
The generated object code is written onto an external storage medium whose reference number is set by the option card. The format of the object code is described in the following Data General publications:

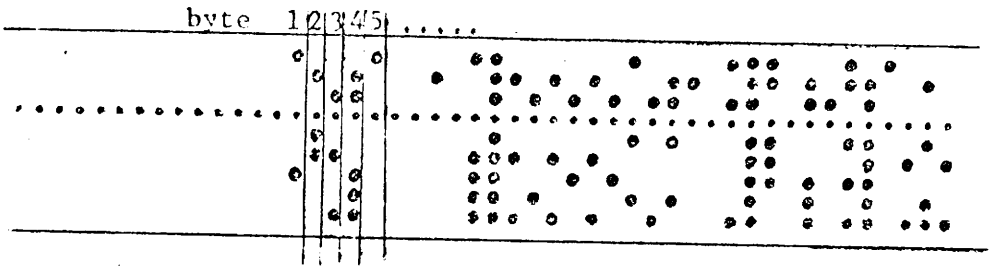
Binary	- 093-000017	(Assembler; Appendix F)
Relocatable	- 093-000040	(Extended Assembler; Appendix A)
	093-000039	(Relocatable Loader; Appendices A-C)

The output format of this assembler is identical to those referenced except that no error blocks are output if errors occur in an assembly.

The contents of the object code file are formatted so that they can be easily copied to a suitable medium, usually paper tape, for loading.

Each word of the file contains one 8-bit paper tape byte, right adjusted in the word. The number of words in each record is variable. Thus, to copy the object code file to paper tape, the program should punch one paper tape byte per word, ignoring end-of-record marks. For example:





APPENDIX A

PROGRAM AND SUBROUTINE DESCRIPTION

This Appendix briefly describes each subprogram in the assembler and is intended for use in conjunction with the listing as an aid to understanding the assembler.

PROGRAM MAIN

MAIN calls the card read program (NEXTCD), which reads the source file in pass one and the scratch file in pass two, and performs a preliminary analysis of the statement type. MAIN then calls the op-code processing routine (OPPROC), the object file output routine (PTAPES), and the listing routine (LISTOT.) This loop is executed for each card on both passes. (A flowchart of MAIN is given at the end of this section.)

SUBROUTINES

ACNUM is called by the op-code processing subroutines to evaluate the accumulator expression field.

ADDREF creates and maintains a cross reference chain (CROSS) for each symbol referenced in an assembly. The pointer to the beginning of the chain is stored in the corresponding symbol type word (SYTYPE).

ADDRES calculates the addressing information portion of memory reference instructions. It evaluates the source card displacement field and the index field. In the absence of the index field, the appropriate index is determined and incorporated into the object instruction. A flowchart of ADDRES is given at the end of this Appendix.)

ADDSYM adds symbols, their values, and types to the symbol table on pass one. On pass two the table is checked for multiply-defined symbols and phase errors.

ADNBLK adds a word to the object code output block. If the location is not sequential or the location type (IWTYPE) changes, the current output block is output and a new one started before the current word is added. ADNBLK also calls the relocation bit routine (RELOC) which enters the relocation bits into the output block.

ADSBLK adds a symbol name in radix 50 code and its value to the symbol output block being processed.

ATOM is called by the expression evaluation routine (EXPRES) to get the next atom of an expression from the card being processed. If the atom is a symbol, it calls the symbol table lookup routine (LOOKUP) to determine its value and type. A number is evaluated (GETNUM) and returned. If the atom is an operator, its mode is determined and returned.

BLOCK is called by OPPROC and processes .BLK pseudo-ops. BLOCK increments the location counter by the value of the expression field. A label, if any, is added to the symbol table (ADDSYM).

CAREND is called when all required op-code fields have been processed. If anything but blanks or a comment field are encountered, an error is flagged.

COMPCT is called at the beginning of each assembly to clean all non-permanent symbols from the symbol table.

CONSS converts the internal code, symbol name character string (LNX) into a single-word symbol name for entry in the symbol table.

CONVRT converts the source card characters from the machine internal code to the assembler internal code. *

CONV50 is called to convert a symbol table name (SYNAME) back to machine internal code for printing in the cross reference table.

CONX converts a symbol table name (SYNAME) to a radix 50 code name for output in symbol blocks.

DEVICE is called by op-code processing routines to evaluate the device field of I/O op-codes.

ENDASM is called by MAIN at the end of pass two. The subroutine outputs any blocks in progress and outputs any symbol blocks required and start blocks. It then flushes the buffer. The cross reference listing routine (XREF) is called and the error tables are listed.

END is called by OPPROC and processes .END, .EOT, and .XPNG statements. If an .END is encountered or generated as the first card of an assembly, the stop routine (FINASM) is called which contains the normal stop statement.

EQUV is called by OPPROC to process equivalence expressions. The expression field is evaluated (EXPRESS), and the value is entered in the symbol table with the label.

ERROR sets the error flag (ERRFLG) true and enters the error identification symbol into the error listing array (NCERS).

EVAL evaluates the number in the character string array (LNX) in the base defined by the radix.

EXPRES evaluates expressions by evaluating each individual atom (ATOM) and performing the indicated operations (IOPER). EXPRES flags as an error any operator or operand out of sequence. All expressions with errors are evaluated as 0. All operations are examined to make sure the final result has a valid amount of relocation. (A flow chart is included at the end of this Appendix.)*

EXTERNL is called to process .ENT, .EXTN, .EXTD statements.

FINASM is called by ENDASM and contains the normal stop for the assembler.

FIND determines which of the options available is being processed on the option card.

FPFMT converts floating point expressions into NOVA/SUPERNOVA floating point format. *

GETDEC is called to evaluate decimal numbers by calling the fetch number string routine (GETNS) and the number evaluation routine EVALN) with a radix of 10.

GETNUM evaluates numbers in expressions. The routine determines which type is being encountered (i.e., single precision, double precision floating point) and the applicable base. The number is evaluated and put in the proper format.

GETSTR transfers an alphanumeric plus period character string from the card array to the string array (LNX), skipping preceding blanks and setting no-load and indirect addressing flags if encountered. Trailing blanks and transparent characters are skipped, with no-load and indirect addressing flags set if encountered.

GETSYM calls GETSTR to load a character string into the string array. If the first nontransparent character is alphabetic or a period, the symbol is converted to a single word value (CONSS) and the column pointer (ICH) is advanced. If the first character is not alphabetic or a period, the column pointer is returned to its original position.

HEADR puts the heading on the top of the listing page.

IATNOL checks the column indicated by the column pointer (ICH) no-load, or indirect addressing symbol. The no-load variable (INOLOD) is set to 8 if # is found, and indirect addressing variable INDRCT is set to 65536 if @ is found. In both cases ICH is incremented by one and checks the next character until neither is found,at which point it returns.

IASCII converts internal code to ASCII. It is used in processing text statements.

IFS is called by OPPROC and processes conditional assembly statements .IFE, .IFN, and ENDC. The routine sets and resets the "don't assemble" flag AKPFLG.

IOPER performs the arithmetic and logical operations required in evaluating expressions.

IPARTY returns IPARTY=1 if the number of bits on is odd and IPARTY=0 if it is even.

LEADR outputs the requested number of blank frames.

LISTOT outputs the listing.

LOAD reads a new set of permanent symbols into the symbol table when LDOPS is specified on the option card. The routine also punches a new set of data cards for the BLOCK DATA subroutine.

LOCATS processes .LOC, .NREL, and .ZREL pseudo-ops, sets the assembly mode indicator (IMODE), and sets the location counters (LOCS) if required.

LOOKUP looks the symbol up in the symbol table and returns the type and value. If the symbol is an ALC type instruction, the symbol is checked for modifiers which are then incorporated into the value.

LOSK performs binary search of the symbol table for the required symbol. If the symbol is found, the array index is returned; if not, the position in the array where it should be is returned.

NEXTCD reads the next card from the appropriate file into the card array (LN) and converts the characters to internal code. The end-of-file flag is set if an end-file is encountered. A read error stops the assembler. On pass-1 the input file is read; on pass-2 the scratch file is read.

NEXTNM gets the next symbol from the source card. It is called in processing .EXTD, .ENT, and .EXTN statements.

NONOLD is called to see if the no load symbol (#) has been encountered illegally.

NOTAT is called to see if the indirect addressing symbol (@) has been encountered illegally.

OCTL converts a number to an array which when printed (FORMAT (6I1)) gives the value of the number in octal.

OPDEF is called by OPPROC and processes .DALC, .DIAC, .DIO, .DIOA, .DMR, .DMRA, and .DUSR pseudo-ops by simulating the processing of the appropriate instruction type.

OPLOOK is called to look up a symbol and set the correct value into ITYPE.

OPPROC is called by the MAIN program to process the statement op-codes. The op-code has already been looked up (OPLOOK) and its type set. Based on this type the appropriate instruction processing routine is called.

OUTLOS adds the local symbol blocks to the intermediate buffer.

OUTNEX adds the normal external blocks to the intermediate buffer.

OUTOLD adds absolute object code blocks to the intermediate buffer. The routine also enters the word count and the checksum into the buffer. OUTOLD contains the ENTRY OUTOLX.

PNBLK transfers the output from the intermediate buffer to the output buffer by calling the object code output routine (PNCH).

PNCH breaks the low order 16 bits of the input word into two 8 bit bytes, right adjusted in the object code buffer. The routine blocks the object code into 40 word records. PNCH contains the ENTRY FLUSH which flushes the buffer.

PTAPES is called by the MAIN program to enter the processed object code into the intermediate buffer.

PTAPET outputs a title block to the intermediate buffer then outputs the block to the object code buffer (PUTBLK).

PUTBLK calculates the checksum and word count of relocatable object code blocks. The routine calls the block output subroutine PNBLK to enter the block into the object code buffer.

PUTOLD adds a word of absolute object code to the intermediate buffer.

RADX is called by OPPROC and processes .RDX pseudo-op code statements, evaluating the expression field and entering this value in the assembly number evaluation base (IRADFX).

RANGE determines if an expression value is within limits set by its mode (single precision, double precision, floating point.).

RELOC adds the relocation bits of an object code word to the intermediate buffer in the position indicated by the relocation bit pointers (IRELO, NRELO).

SETUP interprets the option card and sets the flag and file reference numbers accordingly.

SKPBLK steps through blank or transparent characters in the input array and returns with ICH set to the column in which it finds the first non-transparent character. IATNOL is called for each column to check for a no-load or indirect addressing symbol.

STRBLK initializes a new relocatable output block by zeroing the intermediate buffer, resetting the word count, and setting the first word address and the relocation bit pointers.

STROLD initializes absolute output blocks by resetting the word count and zeroing the intermediate buffer.

TEXTS is called by OPPROX to process .TXT, .TXTO, .TXTE, .TXTF, and .TXTM pseudo-ops. The routine calls the ASCII conversion routine (IASCII) and assigns two characters per word,each character right adjusted in 8 bits with appropriate parity.

TITLE is called by OPPROC to process .TITL pseudo-ops. The routine sets the common title name variable (ITITLE).

TOP3 returns the low order 16 bits of the name variable (NAME).

TYPE1 is called by OPPROX and OPDEF to process arithmetic and logical instructions. The fields acceptable are (label) (opcode) (conditional).

TYPE2 is called by OPPROC and OPDEF to process instructions with ACs. The fields acceptable are (label) (opcode) (AC).

TYPE3 is called by OPPROC and OPDEF to process input/output instructions. The fields acceptable are (label) (opcode) (device code).

TYPE4 is called by OPPROC and OPDEF to process input/output with AC instructions. The fields acceptable are (label) (opcode) (AC) (device code).

TYPE5 is called by OPPROC and OPDEF to process memory reference instructions. The acceptable fields are (label) (opcode) (displacement or address) (index).

TYPE6 is called by OPPROC and OPDEF to process memory reference instructions with an accumulator. The acceptable fields are (label) (opcode) (AC) (displacement or address) (index).

TYPE7 is called by OPPROC and OPDEF to process a user instruction. The acceptable fields are (label) (opcode).

XREFS is called by ENDASM to list the cross reference table.

IRIGHT shifts right with zero fill the contents of a word by the number of bits indicated. *

ILEFT shifts left with zero fill the contents of a word by the number of bits indicated.

IAND performs the logical "and" of two words. *

IOR performs the logical "or" of two words. *

---

* These routines must be modified when changing from IBM to CDC or UNIVAC.

The symbol table in the assembler has room for 1000 symbols. The user may want to make this table either larger or smaller. Modification has two parts:

1. The labeled common SYMTB must be changed, e.g., from

COMMON/SYMTB/SYNAME (1000), SYVALU (1000) SYTYPE (1000), NSYM

to

COMMON/SYMTB/SYNAME (200), SYVALU (200) SYTYPE (200), NSYM

Note: The common specification must be changed in each subroutine having a COMMON/SYMTB/

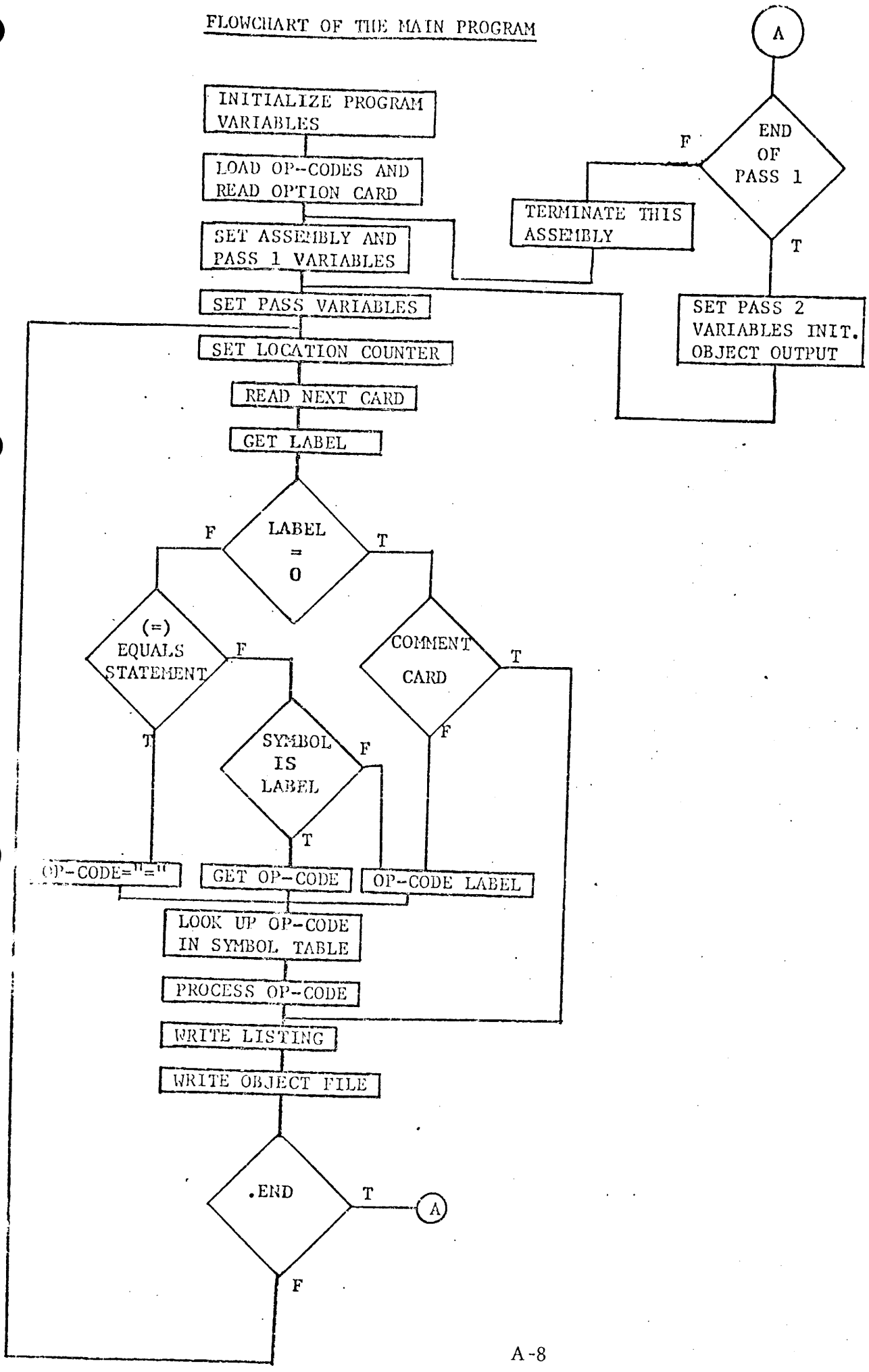
2. Change the NSYMX=1000 statement in the main program to the new specification.

The cross reference table may be altered similiarly by changing the specification of the variable CROSS in blank COMMON and changing the value of NREFX, e.g.,

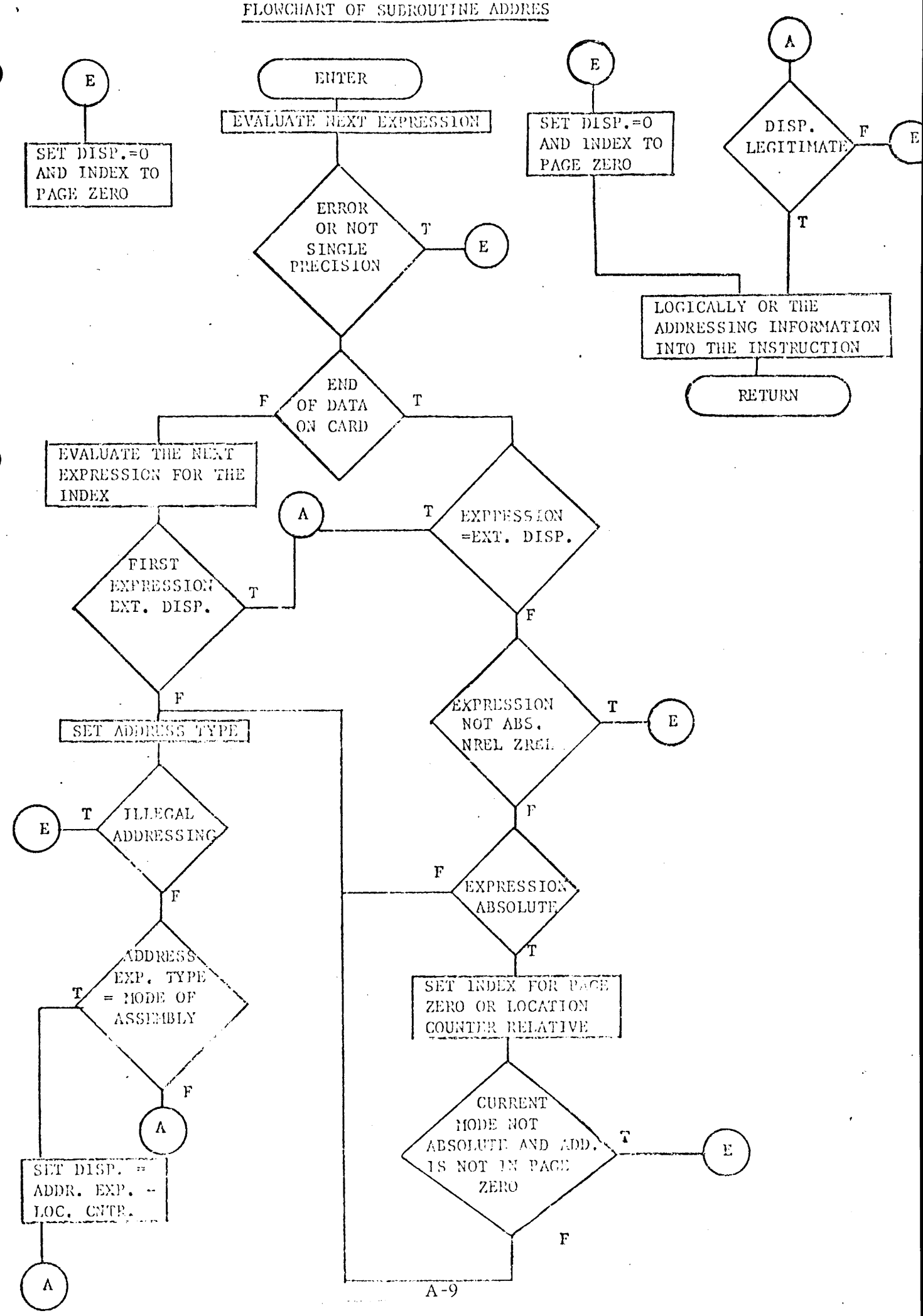
from:	CROSS (1000, 3) and NREFX=1000
to :	CROSS (200, 3) and NREFX=200

Note: The second index in CROSS has a value of 3 which cannot be changed.

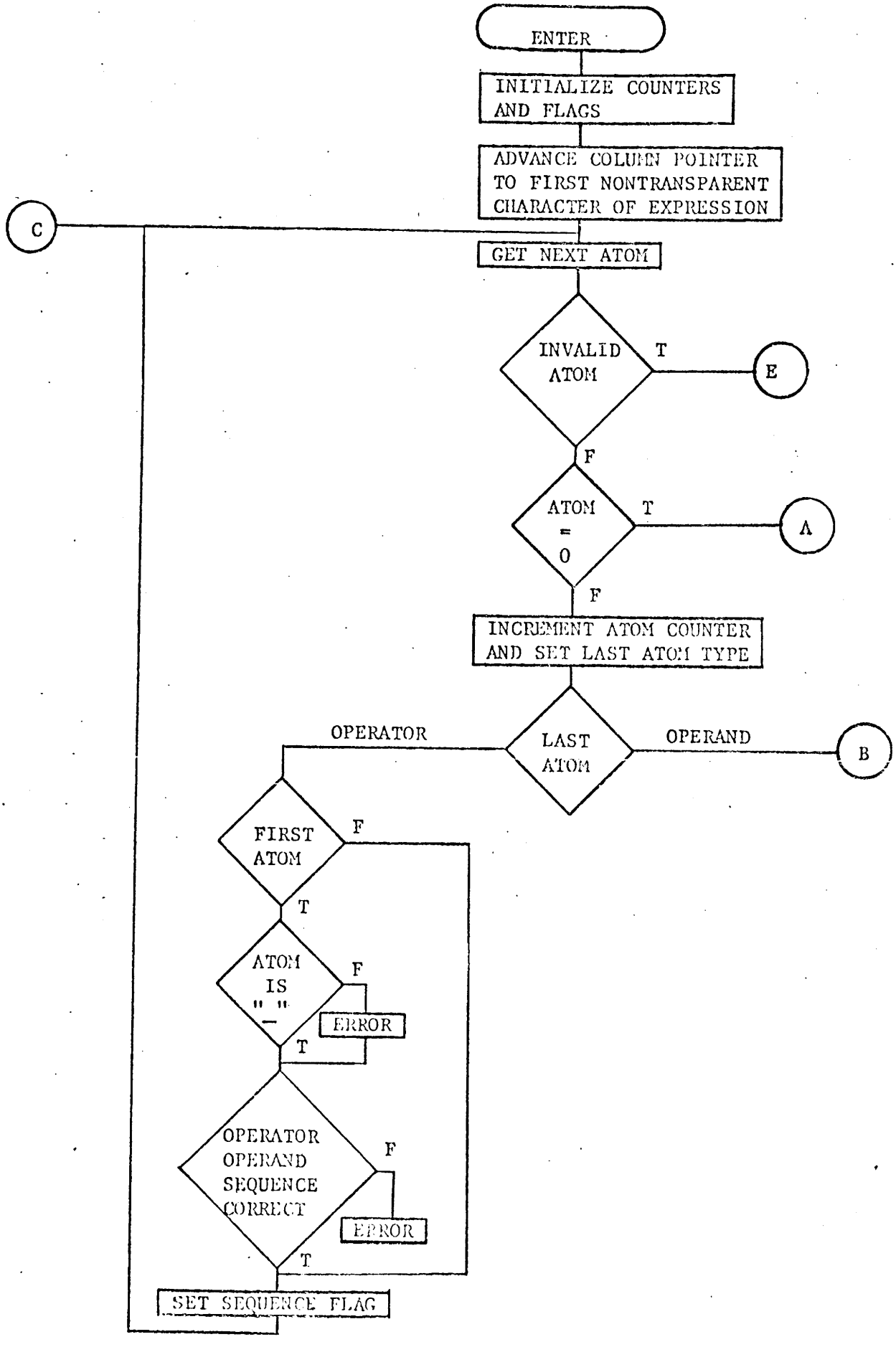
FLOWCHART OF THE MAIN PROGRAM

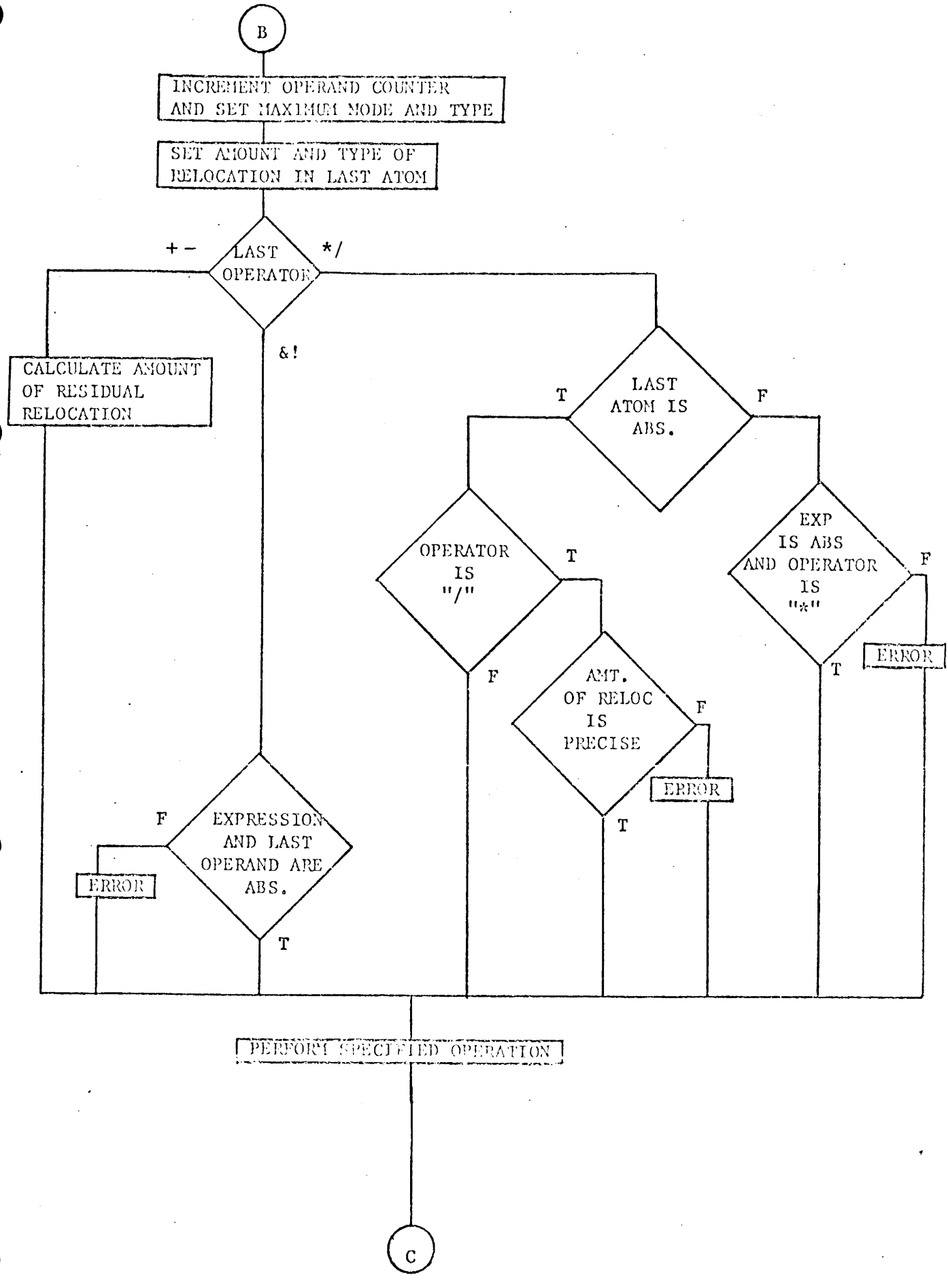


FLOWCHART OF SUBROUTINE ADDRES

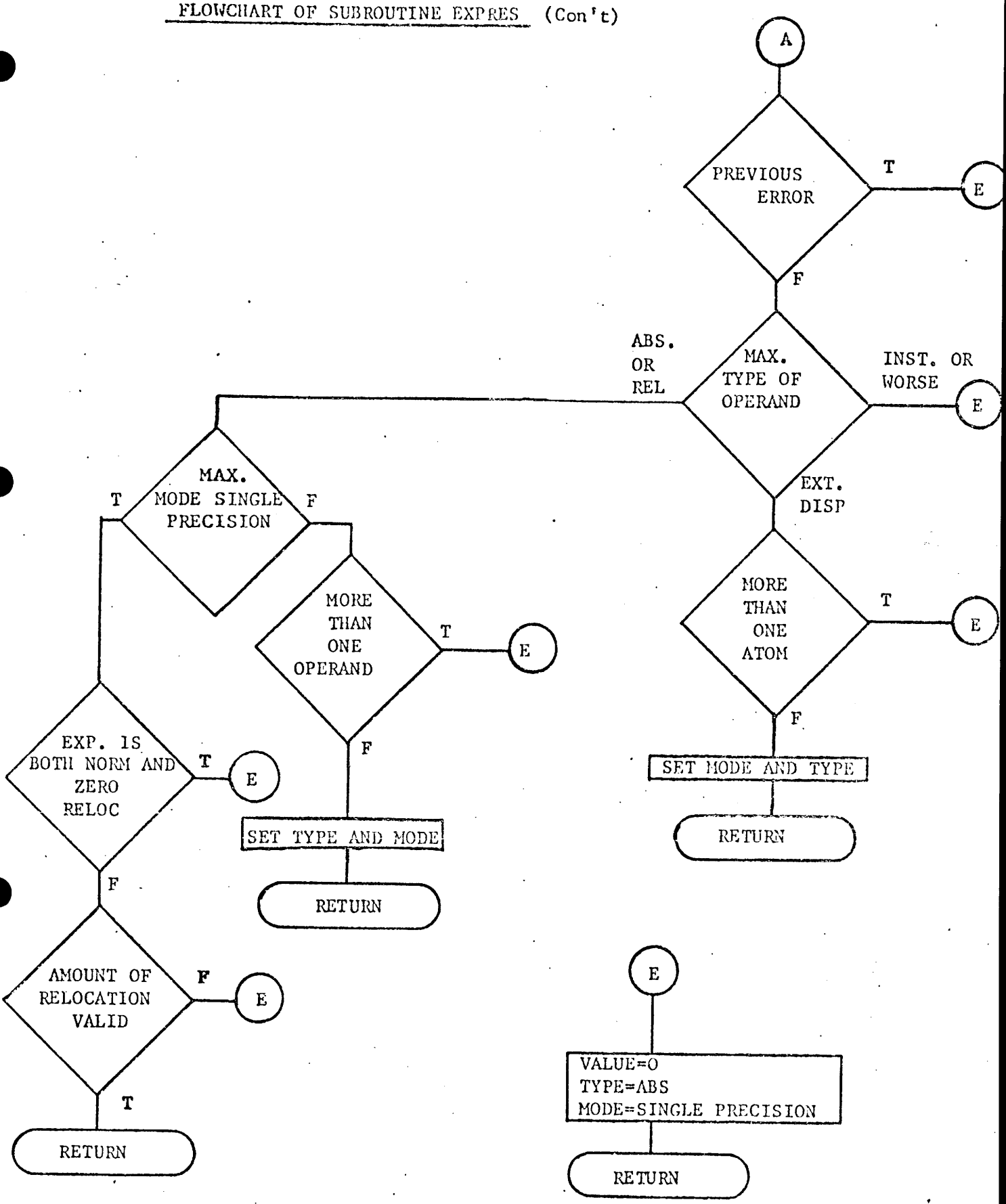


FLOWCHART OF SUBROUTINE EXPRES




B
INCREMENT OPERAND COUNTER AND SET MAXIMUM MODE AND TYPE
SET AMOUNT AND TYPE OF RELOCATION IN LAST ATOM
LAST OPERATOR
+ -
*/
&!
CALCULATE AMOUNT OF RESIDUAL RELOCATION
LAST ATOM IS ABS.
T
F
OPERATOR IS "/"
T
F
AMT. OF RELOC IS PRECISE
F
T
ERROR
EXP IS ABS AND OPERATOR IS "*"
F
T
ERROR
EXPRESSION AND LAST OPERAND ARE ABS.
F
T
ERROR
PERFORM SPECIFIED OPERATION
C

FLOWCHART OF SUBROUTINE EXPRES (Con't)



APPENDIX B

DESCRIPTION OF COMMON BLOCKS

BLANK COMMON

NREF	--	Cross reference table pointer.
ICH	--	ICH points to the next column on the card being processed.
NLNX	--	NLNX indicates the number of characters transferred from the card array (LN) to the symbol array (LNX).
IMODE	--	IMODE indicates the mode (Absolute, Normal Relocatable, Page Zero Relocatable) of the expression being processed.
LOCCNT	--	The value of the current location counter in the mode being processed.
LOCNXT	--	The value to which LOCCNT is set when a new card is begun.
ICORE	--	The number of words of core in the NOVA/SUPERNOVA.
IRADIX	--	The radix of integers to be evaluated by the assembler.
LABEL	--	If a statement has a label it is stored in LABEL.
INDRCT	--	When a @ is encountered, INDRCT is set to 65536.
NLINE	--	The line of output last printed on the output page.
IPAGE	--	The page of output currently being printed.
NCER	--	The number of errors found on the card being processed so far.
NEREF	--	Number of cards having errors.
IOCLAS	--	Indicates the format of the output to be used for the card being processed.

ICARD	--	Source card counter.
NWDS	--	Number of words generated by the instruction processed.
IWTYPE	--	The type of relocation of the word being processed.
IRELO	--	Shift counter used in generating the relocation bits in output blocks.
NRELO	--	Counts the number of relocation bit words generated.
IWDCT	--	Output block word counter.
IADL	--	A location counter used to check for consecutively stored words.
INOLOD	--	INOLOD is set to 8 if a # (no load) character is encountered.
ITITLE	--	Hold the program title generated by the .TITL pseudo-op.
RELASM	--	Set true for relative assemblies and false for absolute assemblies.
XREF	--	Set true if the cross reference table is to be output.
PASS1	--	Set true on the first pass of the assembler.
SKPFLG	--	Set by .IFE, .IFN, and .ENDC statements.
IFMODE	--	Used to test for illegal .IFE, .IFN, and .ENDC statements.
EOFLG	--	Set by an END OF FILE on the source file.
TXTFLG	--	Indicates continuation of text. (not used)
ERRFLG	--	Set true when an error is found on a card.
ANYPCH	--	Used for header and trailer blank paper tape.
CODE	--	Set true by the first non - (.EXT, .ENT, .EXTN, .EXTD, .TITL.) statement.

LOCALS	--	Indicates whether a local symbol block is to be punched.
CROSS	--	Cross reference table.
LN	--	Source card array.
LNX	--	Symbol array.
ISOUR	--	Source card array used for listing.
NEREFS	--	Error, card number array.
NCERS	--	NCERS holds the error symbols for a source card.
IWORD	--	IWORD holds the object code to be formatted and put into the buffer.
IBL	--	IBL is the intermediate output buffer.
LOCS	--	LOCS holds the absolute, normal relocatable and page zero locatable location counters.

COMMON/CDS/

INFILE	--	Source file.
IOFILE	--	Output listing print file.
IPFILE	--	Paper tape output file.
ISCR	--	Scratch file.
ICLAS	--	Holds the relocation symbols.

The rest of the common has the values of certain characters in the assembler internal code.

COMMON/LIMITS/

NSYMX	--	Maximum number of symbols.
NREFX	--	Maximum number of cross reference entries.

NLINEX -- Maximum number of lines on a page.

NCEREX -- Maximum number of errors per card that can be listed.

NEREFX -- Maximum number of card numbers that can be output locating errors.

COMMON/SYMTB/

SYNAME -- Symbol name table.

SYTYPE -- Symbol type table.

SYVALU -- Symbol value table.

NSYM -- Number of symbols in the symbol table.

APPENDIX C

CONTROL VARIABLE VALUES

SYMBOL TYPE

1. Absolute
2. Normally relocatable
3. Page zero relocatable
4. Normal byte relocatable
5. Page zero byte relocatable
6. External displacement
7. UNUSED
8. Expressions
9. Arithmetic and logical instructions (ADD, COM, etc.)
10. Instructions that use one accumulator (READS, INTA, etc.)
11. I/O instructions that may have a device code and no accumulator (NIO, HALT, etc.)
12. I/O instructions that have an accumulator and a device code (DIB, INTA, etc.)
13. Memory reference instructions which use a displacement and may use an address register (JMP, DSZ, etc.)
14. Memory reference instruction which uses an accumulator, displacement, and may use an addressing register (LDA, STA, etc.)
15. USER instructions.
16. Arithmetic and logical instruction modifiers.

17.
18.
19.
20. UNUSED
21.
22.
23.

24. Pseudo ops. (. LOC, . DIO, etc.)

25. Normal external whose last reference was absolute.

26. Normal external whose last reference was normally relocatable.

27. Normal external whose last reference was page zero relocatable.

All permanent symbols have the above values plus 32.

SYMBOL MODE

Mode can refer to an entire expression or to an individual atom.

1. Single precision fixed pt.

2. Double precision fixed pt.

3. Floating point.

4. + (Addition)

5. - (Subtraction)

6. * (Multiplication)

7. / (Division)

8. & (Logical and)

9. ! (Logical or)

IWTYPE

IWTYPE is the relocation type of words to be loaded.

1. Absolute.
2. Normally relocatable.
3. Page zero relocatable.
4. Normally byte relocatable.
5. Page zero byte relocatable.
6. External displacement.

IOCLAS

IOCLAS is the output format to be used in the listing for this instruction.

1. Address and instruction (eg: LDA)
2. Address and instruction (eg: JSR)
3. Value (eg: .LOC, .BLK)
4. (eg: .ENT, .EXTD)

All formats list any errors and the card itself.

IBM Character Codes

CODES

CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	IBM EBCDIC (HEXI-DECIMAL)	IBM 029 PUNCH SET
A	1	65	C1	12-1
B	2	66	C2	12-2
C	3	67	C3	12-3
D	4	68	C4	12-4
E	5	69	C5	12-5
F	6	70	C6	12-6
G	7	71	C7	12-7
H	8	72	C8	12-8
I	9	73	C9	12-9
J	10	74	D1	11-1
K	11	75	D2	11-2
L	12	76	D3	11-3
M	13	77	D4	11-4
N	14	78	D5	11-5
O	15	79	D6	11-6
P	16	80	D7	11-7
Q	17	81	D8	11-8
R	18	82	D9	11-9
S	19	83	E2	0-2
T	20	84	E3	0-3
U	21	85	E4	0-4
V	22	86	E5	0-5
W	23	87	E6	0-6
X	24	88	E7	0-7
Y	25	89	E8	0-8
Z	26	90	E9	0-9

CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	IBM EBCDIC (HEXI-DECIMAL)	IBM 029 PUNCH SET
.	27	46	4B	12-3-8
0	28	48	F0	0
1	29	49	F1	1
2	30	50	F2	2
3	31	51	F3	3
4	32	52	F4	4
5	33	53	F5	5
6	34	54	F6	6
7	35	55	F7	7
8	36	56	F8	8
9	37	57	F9	9
+	38	43	4E	12-6-8
-	39	45	60	11
*	40	42	5C	11-4-8
/	41	47	61	0-1
&	42	38	50	12
!	43	33	5A	11-2-8
'	44	32	40	———
)	45	41	6B	0-3-8
;	46	34	7F	7-8
@	47	64	7C	4-8
#	48	35	7B	3-8
;	49	59	5E	11-6-8
:	50	58	7A	2-8
<	51	60	4C	12-4-8
>	52	62	6E	0-6-8
=	63	61	7E	6-8

CDC Character Codes

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	CDC PRINT CHARACTER	CDC CONSOLE DISPLAY CODE (DECIMAL)	IBM 029 PUNCH SET	IBM 029 CHARACTER
A	1	65	A	1	12-1	A
B	2	66	B	2	12-2	B
C	3	67	C	3	12-3	C
D	4	68	D	4	12-4	D
E	5	69	E	5	12-5	E
F	6	70	F	6	12-6	F
G	7	71	G	7	12-7	G
H	8	72	H	8	12-8	H
I	9	73	I	9	12-9	I
J	10	74	J	10	11-1	J
K	11	75	K	11	11-2	K
L	12	76	L	12	11-3	L
M	13	77	M	13	11-4	M
N	14	78	N	14	11-5	N
O	15	79	O	15	11-6	O
P	16	80	P	16	11-7	P
Q	17	81	Q	17	11-8	Q
R	18	82	R	18	11-9	R
S	19	83	S	19	0-2	S
T	20	84	T	20	0-3	T
U	21	85	U	21	0-4	U
V	22	86	V	22	0-5	V
W	23	87	W	23	0-6	W
X	24	88	X	24	0-7	X
Y	25	89	Y	25	0-8	Y
Z	26	90	Z	26	0-9	Z

CDC Character Codes (Continued)

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	CDC PRINT CHARACTER	CDC CONSOLE DISPLAY CODE (DECIMAL)	IBM 029 PUNCH SET	IBM 029 CHARACTER
.	27	46	.	47	12-3-8	.
0	28	48	0	27	0	0
1	29	49	1	28	1	1
2	30	50	2	29	2	2
3	31	51	3	30	3	3
4	32	52	4	31	4	4
5	33	53	5	32	5	5
6	34	54	6	33	6	6
7	35	55	7	34	7	7
8	36	56	8	35	8	8
9	37	57	9	36	9	9
+	38	43	+	37	12	&
-	39	45	-	38	11	-
*	40	42	*	39	11-4-8	*
/	41	47	/	40	0-1	/
&	42	38	≤	58	12-0	None
!	43	33	∨	54	11-0	None
Blank	44	32	Blank	45	———	No Punch
,	45	44	,	46	8-3	,
"	46	34	≤	50	8-5	' (H)
@	47	64	≠	51	8-4	@
#	48	35	:	52	8-2	:
;	49	59	;	63	12-8-7	¦ (Y)
:	50	58	$	43	11-3-8	$
<	51	60	(	41	0-4-8	%
>	52	62	)	42	12-4-8	<
=	63	61	=	44	3-8	#

UNIVAC Character Codes

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	UNIVAC 1004 PRINT CHARACTER	UNIVAC 1108 INTERNAL CODE	IBM 029 PUNCH SET	IBM 029 CHARACTER
A	1	65	A	6	12-1	A
B	2	66	B	7	12-2	B
C	3	67	C	8	12-3	C
D	4	68	D	9	12-4	D
E	5	69	E	10	12-5	E
F	6	70	F	11	12-6	F
G	7	71	G	12	12-7	G
H	8	72	H	13	12-8	H
I	9	73	I	14	12-9	I
J	10	74	J	15	11-1	J
K	11	75	K	16	11-2	K
L	12	76	L	17	11-3	L
M	13	77	M	18	11-4	M
N	14	78	N	19	11-5	N
O	15	79	O	20	11-6	O
P	16	80	P	21	11-7	P
Q	17	81	Q	22	11-8	Q
R	18	82	R	23	11-9	R
S	19	83	S	24	0-2	S
T	20	84	T	25	0-3	T
U	21	85	U	26	0-4	U
V	22	86	V	27	0-5	V
W	23	87	W	28	0-6	W
X	24	88	X	29	0-7	X
Y	25	89	Y	30	0-8	Y
Z	26	90	Z	31	0-9	Z

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	CDC PRINT CHARACTER	CDC CONSOLE DISPLAY CODE (DECIMAL)	IBM 029 PUNCH SET	IBM 029 CHARACTER
.	27	46	.	47	12-3-8	.
0	28	48	0	27	0	0
1	29	49	1	28	1	1
2	30	50	2	29	2	2
3	31	51	3	30	3	3
4	32	52	4	31	4	4
5	33	53	5	32	5	5
6	34	54	6	33	6	6
7	35	55	7	34	7	7
8	36	56	8	35	8	8
9	37	57	9	36	9	9
+	38	43	+	37	12	&
-	39	45	-	38	11	-
*	40	42	*	39	11-4-8	*
/	41	47	/	40	0-1	/
&	42	38	≤	58	12-0	None
!	43	33	∨	54	11-0	None
Blank	44	32	Blank	45	———	No Punch
,	45	44	,	46	8-3	,
"	46	34	≤	50	8-5	' (H)
@	47	64	≠	51	8-4	@
#	48	35	:	52	8-2	:
;	49	59	;	63	12-8-7	| (Y)
:	50	58	$	43	11-3-8	$
<	51	60	(	41	0-4-8	%
>	52	62	)	42	12-4-8	<
=	63	61	=	44	3-8	#

UNIVAC Character Codes

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	UNIVAC 1004 PRINT CHARACTER	UNIVAC 1108 INTERNAL CODE	IBM 029 PUNCH SET	IBM 029 CHARACTER
A	1	65	A	6	12-1	A
B	2	66	B	7	12-2	B
C	3	67	C	8	12-3	C
D	4	68	D	9	12-4	D
E	5	69	E	10	12-5	E
F	6	70	F	11	12-6	F
G	7	71	G	12	12-7	G
H	8	72	H	13	12-8	H
I	9	73	I	14	12-9	I
J	10	74	J	15	11-1	J
K	11	75	K	16	11-2	K
L	12	76	L	17	11-3	L
M	13	77	M	18	11-4	M
N	14	78	N	19	11-5	N
O	15	79	O	20	11-6	O
P	16	80	P	21	11-7	P
Q	17	81	Q	22	11-8	Q
R	18	82	R	23	11-9	R
S	19	83	S	24	0-2	S
T	20	84	T	25	0-3	T
U	21	85	U	26	0-4	U
V	22	86	V	27	0-5	V
W	23	87	W	28	0-6	W
X	24	88	X	29	0-7	X
Y	25	89	Y	30	0-8	Y
Z	26	90	Z	31	0-9	Z

UNIVAC Character Codes (Continued)

CODES

DATA GENERAL ASSEMBLER CHARACTER	ASSEMBLER INTERNAL CODE (DECIMAL)	ASCII (DECIMAL)	UNIVAC 1004 PRINT CHARACTER	UNIVAC 1108 INTERNAL CODE	IBM 029 PUNCH SET	IBM 029 CHARACTER
.	27	46	.	61	12-3-8	.
0	28	48	0	48	0	0
1	29	49	1	49	1	1
2	30	50	2	50	2	2
3	31	51	3	51	3	3
4	32	52	4	52	4	4
5	33	53	5	53	5	5
6	34	54	6	54	6	6
7	35	55	7	55	7	7
8	36	56	8	56	8	8
9	37	57	9	57	9	9
+	38	43	+	34	12	&
-	39	45	-	33	11	-
*	40	42	*	40	11-4-8	*
/	41	47	/	60	0-1	/
&	42	38	?	44	12-0	None
!	43	33	!	45	11-0	None
↑	44	32	Blank	5	———	No Punch
,	45	44	,	46	8-3	,
"	46	34	:	43	8-5	' (H)
@	47	64	'	58	8-4	@
#	48	35	&	38	8-2	:
;	49	59	#	3	12-8-7	| (Y)
:	50	58	$	39	11-3-8	$
<	51	60	(	41	0-4-8	%
>	52	62	)	32	12-4-8	<
=	63	61	=	36	3-8	#

APPENDIX E

ERROR CODES

FLAG	ERROR TYPE	EXAMPLES AND COMMENTS
A	Address Error	LDA 0,400 ISZ .+317 ;ADDRESS OUTSIDE RANGE. RANGE ;MAY BE FOR ABSOLUTE ZERO RE- ;LOCATABLE, OR NORMAL RELOCATABLE.
B	Bad Character	LA$L; LDA 1,23 ;$ NOT PERMITTED.
C	Colon Error	A+2: ;NO EXPRESSION PERMITTED BEFORE ;COLON.
D	Radix Error	.RDX 12 ;RADIX 12 NOT PERMITTED.
E	Equal Error	REG= 3+B ;B IS UNDEFINED.
F	Format Error	ADD 2 ;NEED AT LEAST TWO OPERANDS.
G	Symbol Dec- laration Error	.EXTN S5 ;S5 NOT DEFINED AS .ENT IN SOME ;OTHER PROGRAM.
I	Input Error	;PARITY CHECKED ON INPUT AND ;SOME CHARACTER WAS IN ERROR.
K	Conditional Assembly Error	;.IFE/.IFN PSEUDO-OP EXPRESSION ;NOT EVALUABLE IN PASS 1 OR ;.IFE/.IFN NESTED WITH PREVIOUS ;CONDITION ASSEMBLY STATEMENT.
L	.LOC Error	.LOC -1 ;BIT 0 SET.
M	Multiply- defined Symbol	A: 3 A: 4 ;SYMBOL MAY APPEAR ONLY ONCE IN ;LABEL FIELD.
N	Number Error	C77: 7A ;NO LETTERS PERMITTED IN A NUMBER.
O	Field Over- flow	LDA 4,.LOC ;NO REGISTER 4.

FLAG	ERROR TYPE	EXAMPLES AND COMMENTS	
P	Phase Error		;VALUE OF A SYMBOL IN PASS 1 DIF- ;FERS FROM THAT OF PASS 2.
Q	Questionable Line	.+. END	
R	Expression Error	J: C+F	;UNCANCELED MIX OF PAGE ZERO AND ;NREL SYMBOLS.
S	Symbol Table Overflow		;MEMORY CAPACITY FOR A GIVEN ;MACHINE HAS BEEN REACHED.
T	Error in Table Pseudo-op	14+. XPNG	;NO EXPRESSION BEFORE A TABLE ;PSEUDO-OP.
U	Undefined Symbol		;A SYMBOL IN OPERAND FIELD WAS ;NEVER DEFINED.
X	Text Error	LET: "C3 3+. TXT	;ONLY ONE CHARACTER IN " ATOM. ;PSEUDO-OP.
Z	Illegal Symbol in Expression	STA 2, 6D+3	;DOUBLE PRECISION NO. USED IN ;EXPRESSION.

INDEX

ABASM option 3-1

ABSOLUTE assembly
- loading output 1-1
- option 3-1

ASCII code App. D

ASSEMBLER
- common blocks App. B
- control variable values App. C
- programs of App. A
- relation to other DGC assemblers 3-5
- written in FORTRAN 1-1

ASSEMBLY
- card input deck Chapt. 3
- features 1-1
- language 3-4
- tape of 2-1

CDC 6600
- assembler routine modification A-6
- conversion codes D-2, D-3
- input deck for 3-6
- program tape 2-1

CODE conversion
- CDC D-2, D-3
- IBM D-1
- punches App. D
- UNIVAC D-4, D-5

COMMON blocks App. B

CONTROL variables App. B

CROSS reference table
- changing size of A-7
- format 4-3
- suppression of 3-2

.EOT pseudo-op 3-4

ERROR
- code meanings E-1, E-2
- codes in listing 4-1
- list with card numbers 4-3

FILE
- assignment 3-3
- IN 3-1, 3-3
- of option card 3-1,
- OUT 3-1, 3-3
- PT 3-1, 3-3
- SCR 3-1, 3-3

FLOWCHART
- ADDRES A-9
- EXPRES A-10
- MAIN A-8

FORTRAN 1-1

IBM 360-OS
- conversion codes D-1
- input deck 3-6
- program tape for 2-1

IN file 3-1, 3-3

INPUT
- CDC 6600 deck 3-6
- file 3-3
- IBM 360-OS deck 3-6
- medium for 3-4
- normal assembly deck 3-4, 3-5
- op-code change deck 3-4, 3-5
- option cards for 3-1
- source code deck 3-4
- UNIVAC 1108 deck 3-7

JOB control cards 2-1

LDOPS option 3-2

LISTING
errors 4-3
source text 4-1, 4-2
symbol cross reference 4-3

LOADING output 4-4

LOCAL symbol table 3-2

LOCLS option 3-2

MACHINE requirements 2-1

NXREF option 3-2

OBJECT code
copy for loading 2-1, 4-4
file 3-3
format 4-4
suppression of 3-2

OP-CODES
deck format 3-4
modification of 1-1, 3-2

OPTION card
default 3-1, 3-3
examples 3-3
file for 3-1
file-assignment field 3-3
format 3-1
general content 1-1
option field 3-1

OPTIONS
ABASM 3-1
LDOPS 3-2
LOCLS 3-2
NXREF 3-2
PTAPE 3-2
RLASM 3-1

OUT file 3-1, 3-3

OUTPUT
listing
errors 4-1, 4-3
file 3-3
source text 4-1, 4-2
symbol cross reference 4-3
object code
copying 2-1, 4-4
file for 3-3
format 4-4

PAPER tape output 4-4

PROGRAM units of assembler App. A

PT file 3-1, 3-3

PTAPE option 3-2

PUNCH codes App. D

RELOCATABLE assembly
loading output 1-1
option 3-1

RELOCATION symbols 4-1

RLASM option 3-1

SCR (scratch) file 3-1, 3-3

SOURCE code
deck 3-4, 3-5
text listing 4-1, 4-2

SUBROUTINES of assembler App. A

SYMBOL
cross reference table 4-3
local symbol table 3-2
permanent 3-2
replacement of permanent 3-2
table size modification A-7

UNIVAC 1108
assembler routine modifications A-6
conversion codes D-4, D-5
input deck 3-7
program tape for 2-1

The programs for the Assembler for the IBM 360, CDC 6600, and UNIVAC 1108 are offered on an "as is" basis. Data General Corporation assumes no responsibility for maintenance of the program as it is described, for any necessary changes to tailor it to a particular software or hardware configuration, or for preparation of Job Control cards or other control information necessary for execution on a particular configuration. Since the programs are written in FORTRAN IV, any user alteration should be comparatively simple.

CONTENTS

Chapter 1	-	GENERAL DESCRIPTION	1-1
Chapter 2	-	MACHINE REQUIREMENTS	2-1
Chapter 3	-	INPUT	3-1
		Option Card Format	3-1
		Option Fields	3-1
		File-Assignment Fields	3-3
		Option Card Examples	3-3
		Op-code Deck	3-4
		Source Code Deck	3-5
		Examples of Deck Setups	3-6
		Normal Assembly	3-6
		Assembly with New Op-codes	3-6
		IBM 360-OS	3-7
		CDC 6600	3-7
		UNIVAC 1108	3-8
Chapter 4	-	OUTPUT	4-1
		Listing	4-1
		Source Text Listing	4-1
		Sample Source Text Listing	4-2
		Cross Reference Table	4-3
		Type of Assembly and Error List	4-3
		Sample Cross Reference and Error Listings	4-3
		Object Code	4-4
APPENDIX A	-	Program and Subroutine Description	
APPENDIX B	-	Description of Common Blocks	
APPENDIX C	-	Control Variable Values	
APPENDIX D	-	Character Code Conversions	
APPENDIX E	-	Error Codes	