3100
3200
3300

Control Data®

Basic Assembler
Reference Manual

Any comments concerning this manual should be addressed to:

CONTROL DATA CORPORATION
Documentation Department
3145 PORTER DRIVE
PALO ALTO, CALIFORNIA

September, 1964
Pub. No. 60057100

INTRODUCTION

BASIC Assembly provides an efficient method of putting machine language programs into production and may be used with any configuration as a part of the comprehensive BASIC system. The language includes mnemonic operation codes, symbolic addressing techniques and a set of pseudo operations.

BASIC Assembly operates as a two-pass assembler with a symbol table retained in storage between passes. During the first pass, source input is read, values are assigned to location symbols, a check is made for doubly defined symbols and the values are stored in the symbol table. During the second pass, source input is read, the symbol table is searched for address terms, binary equivalents are assembled for the source code line, and listable or binary output is produced unless suppressed. If both list and binary output are assigned to the same physical unit, a third pass is necessary to read the source input and produce binary output.

CONTENTS

3200 COMPASS SYSTEM CODING FORM

CONTROL DATA CORPORATION

PROGRAM	NAME
ROUTINE	PAGE
	DATE

LOCN	OPERATION, MODIFIERS	ADDRESS FIELD	COMMENTS	IDENT
1–8	10–19	20–40	41–72	73–80

CDC 366

BASIC LANGUAGE 1

CODING FORMAT

The BASIC assembly format described here is used for all BASIC assembly instructions, symbolic or octal machine instructions and pseudo instructions.

CODING FIELDS

A code line is divided into five fields:

The location field begins with column 1 and ends with column 8. Column 9 must be blank for BASIC assembly code lines.

The operation field begins with column 10.

The address field begins with the first non-blank column following the operation field or column 20; it must begin before column 41 and is terminated with the first blank or column 72, whichever occurs first.

The comments field begins with column 41 or the first non-blank column following the address field and ends with column 72. When machine instructions have blank address fields, comments may begin after the first blank following the operation field.

The identification field, which is not printed on the output listing, appears in column 73 to 80.

The operation and address field may each have several subfields.

OPERATION SUBFIELDS

The operation field has one or more subfields, separated by commas. The first subfield, the operation code, specifies the operation to be performed. Succeeding subfields are modifiers specifically related to the operation code. Modifiers indicate indirect addressing, sign extension, input/output options, character addressing and jump conditions.

ADDRESS SUBFIELDS

The address field has one to three subfields, separated by commas. Instructions have implied subfields. If the address field is blank, each implicit subfield assumes the value zero. An individual subfield may be skipped and assigned the value zero by giving only its trailing comma or, if it is the last subfield in the address field, by omitting both the value and the preceding comma.

subfield	
m or n	a word address which specifies the location of full word data
y	word data
r or s	a character address which specifies the location of partial word or character data
b	specifies indexing or directs usage of the index register
c	the character to be searched
l	the length of a character field to be moved
v	a register file location or character data
ch	the number of the input/output channel
x	function code or comparison mask for input/output instructions
i	an interval for search instructions

TYPICAL MACHINE CODE

Machine instructions have one to three address subfields, separated by commas, or they may be blank. Typical formats for address subfields are noted below:

operation subfields	address subfields
TIA	b
TMA	v
ECHA, S	r
ENA	y
AZJ, EQ	m
INAC	ch
PAUS	x
LDA	m, b
LACH	r, l
ENI	y, b
TMI	v, b
MEQ	m, i
CON	x, ch
MOVE	l, r, s
SRCE	c, r, s
INPC	ch, r, s
INPW	ch, m, n
SLS	(blank) or remarks

TYPICAL PSEUDO OPERATION CODE

Pseudo instructions have one or two address subfields, separated by a comma, or they may be blank. Individual subfields are defined in Chapter 2. Typical formats:

location field	operation subfields	address subfields
	BSS	m
	BSS, C	m
	OCT	m
	DEC	d
	DECD	d
	BCD	n, (4n characters)
	BCD, C	n, (n characters)
	END	m
a	EQU	m
b	EQU, C	r
	ORGR	m
	LIST	(blank)
	NOLIST	(blank)
	REM	
	EJECT	(blank)
	SPACE	(blank) or m

CODING ELEMENTS

The following elements of code are placed in operation subfields: machine or pseudo operation mnemonics, and mnemonic modifiers. Address subfields may contain numbers, symbols, a single asterisk or a combination of two of the foregoing, a double asterisk or remarks.

MACHINE INSTRUCTION

Mnemonics	Octal		Mnemonics	Octal	
HLT	00	Unconditional Stop	SBAQ	33	Subtract from AQ
SJ1-6		Selective Jump 1-6	RAD	34	Replace Add
RTJ		Return Jump	SSA	35	Selectively Set A
UJP	01	Unconditional Jump	SCA	36	Selectively Complement A
IJI	02	Index Jump, Incremental	LPA	37	Logical Product A
IJD		Index Jump, Decremental	STA	40	Store A
AZJ	03	Compare A with Zero	STQ	41	Store Q
AQJ		Compare A with Q	SACH	42	Store A, Character
ASE	04	Skip if (A) = y	SQCH	43	Store Q, Character
QSE		Skip if (Q) = y	SWA	44	Store Word Address
ISE		Skip if (B^b) = y	STAQ	45	Store AQ
ASG	05	Skip if (A) ≧ y	SCHA	46	Store Character Address
QSG		Skip if (Q) ≧ y	STI	47	Store Index
ISG		Skip if (B^b) ≧ y	MUA	50	Multiply A
MEQ	06	Masked Equality Search	DVA	51	Divide A
MTH	07	Masked Threshold Search	CPR	52	Compare
SSH	10	Storage Shift	---	53	Inter-Register Transfers, 24 Bit
ISI		Index Skip, Incremental	LDI	54	Load Index
ISD		Index Skip, Decremental	---	55	Inter-Register Transfers, 48 Bit
ECHA	11	Enter A, Character Address	MUAQ	56	Multiply AQ
SHA	12	Shift A	DVAQ	57	Divide AQ
SHQ		Shift Q	FAD	60	Floating Point Add
SHAQ	13	Shift AQ	FSB	61	Floating Point Subtract
SCAQ		Scale AQ	FMU	62	Floating Point Multiply
ENA	14	Enter A	FDV	63	Floating Point Divide
ENQ		Enter Q	LDE	64	Load E
ENI		Enter Index	STE	65	Store E
INA	15	Increase A	ADE	66	Add to (E)
INQ		Increase Q	SBE	67	Subtract from (E)
INI		Increase Index	SFE	70	Shift E
XOA	16	Exclusive OR of A and y	EZJ		E Zero Jump
XOQ		Exclusive OR of Q and y	EOJ		E Overflow Jump
XOI		Exclusive OR of Index and y	SET		Set D Register
ANA	17	AND of A and y	SRCE	71	Search Character Equality
ANQ		AND of Q and y	SRCN		Search Character Inequality
ANI		AND of Index and y	MOVE	72	Move Data
LDA	20	Load A	INPC	73	Input, Character Block to Storage
LDQ	21	Load Q	INAC		Input, Character to A
LACH	22	Load A, Character	INPW	74	Input, Word Block to Storage
LQCH	23	Load Q, Character	INAW		Input, Word to A
LCA	24	Load Complement A	OUTC	75	Output, Character Block from Storage
LDAQ	25	Load AQ	OTAC		Output, Character from A
LCAQ	26	Load Complement AQ	OUTW	76	Output, Word Block from Storage
LDL	27	Load A Logical	OTAW		Output, Word from A
ADA	30	Add to A	---	77	Sense, Select, Interrupt and Control Functions
SBA	31	Subtract from A			
ADAQ	32	Add to AQ			

PSEUDO INSTRUCTION MNEMONICS

BCD	Insert BCD characters
BSS	Reserve blocks of storage
DEC	Insert single precision decimal constants
DECD	Insert double precision decimal constants
END	Specify the end of a program
EQU	Equate an undefined symbol to a defined word address symbol
LIST	Resume output listing
NOLIST	Suppress output listing
OCT	Insert octal constants
ORGR	Set location counter
REM	Insert remarks on the output listing
EJECT	Eject page of output listing
SPACE	Space output listing
NOP	No operation
IDENT	Program identification

MODIFIERS

EQ	Equal
NE	Not equal
GE	Greater than or equal
LT	Less than
I	Indirect addressing
S	Extend sign of operand to 24 bits
INT	Interrupt on completion
NC	No internal conversion
B	Backward read or write
H	Half assembly or disassembly (12-24)
N	No assembly or disassembly
C	Assign character address
A	Internal BCD alteration

NUMBERS

A decimal number is represented by decimal digits only in an address subfield. An octal number is represented by octal digits suffixed by a B in the address subfield.

Examples	Result
12370B	12370
-12370B	65407
2229	04265
-2222	73512

SYMBOLS

A symbol is a combination of alphabetic (A to Z), numeric (1 to 9), or special (a period) characters up to six in length. Each symbol must begin with a letter of the alphabet. Imbedded blanks are illegal.

Examples:

Legal	Illegal
A12345	123456
ABLE	.23456
BAKER	B 1.2
B123.3	

ASTERISK

If the keypunch character, *, appears in column one of the card, the entire card is treated as remarks. In an address subfield, an asterisk implies self-reference.

EXPRESSION

An expression is a number, a symbol, an asterisk, or two of these joined by a plus or minus sign. If S represents a symbol and N represents a number, the following combinations are permitted:

S±S	N±N	*±*
S±N	N±*	
S±*		

DOUBLE ASTERISK

When two consecutive asterisk keypunch characters, ** , are used, one bits are inserted into a given size subfield.

REMARKS

Remarks are any combination of keypunch characters.

WORD AND CHARACTER ADDRESSING

Address subfields may contain any legal code element which results in either a character or word address. A 24-bit machine word is referenced by a 15-bit word address. A 6-bit portion (character) of a machine word is referenced by a 17-bit address; the two extra bits indicate character position.

Character is a 6-bit configuration; each machine word contains four characters.

Character position is the place within a word occupied by a character; a character may occupy position 0, 1, 2 or 3 as follows:

Bits	23 18	17 12	11 6	5 0
Character Position	0	1	2	3

Word address is a coding element for address subfields which results in a 15-bit value. The address represents the location of a 24-bit machine word or word data.

Character address is a coding element for address subfields which results in a 17-bit value. The left 15 bits represent the location of the word containing the character. The other 2 bits represent the position of the character within the word. The 17-bit values indicate the position of the character as follows:

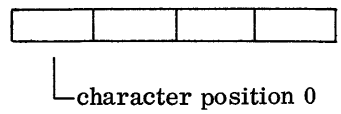
xxxxxxxxxxxxxxx00	character position 0
xxxxxxxxxxxxxxx01	character position 1
xxxxxxxxxxxxxxx10	character position 2
xxxxxxxxxxxxxxx11	character position 3

The following are binary address of characters:

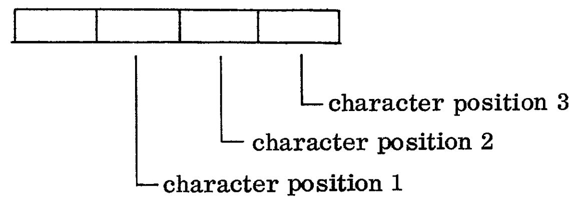
00000000100000000
00000000100000001
00000000100000010
00000000100000011

In the first example, the left 15 bits indicate location 00100_8. The last 2 bits indicate position zero.

word 00100_8



The next examples indicate location 00100_8, character positions 1, 2 and 3.



Word Instruction Machine instructions which require word addresses are indicated by m, n or y subfields in Table 1. Word addresses are evaluated modulo 2^{15}.

Character Instruction Machine instructions which require character addresses are indicated by an r or s subfield in Table 1. Character addresses are evaluated modulo 2^{17}.

WORD ADDRESS REPRESENTATION

Word addresses (15-bit values) formed in m or n subfields result in the following address types:

coding elements	address type
decimal	octal
octal	octal
symbol	relative or octal
expression	relative or octal
*	relative or octal
**	special

Examples:

LDA m instruction

coding elements		address type
LDA	1908	relative
LDA	1772B	octal
LDA	ABLE	relative
LDA	ABLE+1772B	relative
LDA	*	relative
LDA	**	special

DECIMAL ADDRESS

A decimal number is converted to an octal number right justified in a 15-bit field with sign extended throughout the field.

LDA	m	instruction
LDA	1299	relative address
LDA	02423	result in octal

OCTAL ADDRESS

An octal number is right justified in a 15-bit field with sign extended throughout the 15-bit field.

LDA	m	instruction
LDA	1277B	octal address
LDA	01277	result in octal

SPECIAL ADDRESS

When an asterisk appears in a word address subfield, the 15-bit current value of the location counter is assigned.

LDA	m	instruction
LDA	*	relative address (value of location counter is 00100_8)
LDA	00100	result in octal

When a double asterisk appears in a word address subfield, a 15-bit value of all one bits is assigned.

LDA	m	instruction
LDA	**	special address
LDA	77777	result in octal

RELATIVE ADDRESS

A symbol in an m or n address subfield must be defined elsewhere as a location symbol. The 15-bit value assigned to that location symbol is used as the value of the symbol when it appears in an address field.

LDA	m	instruction
LDA	ABLE	symbolic tag (ABLE previously assigned value 01277_8)
LDA	01277	result in octal

If the symbol were previously assigned a 17-bit value, the right 2 bits are lost; an error diagnostic is given if the 2 bits are non-zero, and the remaining 15-bit value is assigned to this symbol.

Coding

ABLE	BSS, C	5B
	.	
	.	
	LDA	ABLE

The ABLE value assigned is 00000000100000010 or 00100_8, character position 2. The result is:

T	LDA	00100

(error code)

An expression results in the addition or subtraction of two 15-bit values. In the following examples, ABLE is assigned the value 01234_8 and the current value of the location counter is 00111_8.

Examples:

LDA m instruction

Coding		Result	
LDA	ABLE+12	LDA	01250_8
LDA	ABLE-12	LDA	01220_8
LDA	ABLE+*	LDA	01345_8
LDA	ABLE-*	LDA	01123_8
LDA	ABLE+12B	LDA	01246_8
LDA	ABLE-12B	LDA	01222_8
LDA	* + *	LDA	00222_8
LDA	* - *	LDA	00000_8
LDA	129-12	LDA	00165_8

CHARACTER ADDRESS REPRESENTATION

Character addresses (17-bit values) are formed for elements of code placed in r or s subfields or in address subfields of instructions with a C modifier. The coding elements result in the following address types; character address values are evaluated module 2^{17}.

coding elements	address type
symbol	relative
expression	relative
*	relative
**	special
decimal value 2^{17}	
octal value 2^{17}	

Examples:

LACH	r	instruction
coding elements		address type
LACH	ABLE	relative
LACH	ABLE+1772B	relative
LACH	*	special
LACH	**	special
BSS, C	m	instruction
BSS, C	ABLE	relative

SPECIAL ADDRESS

An asterisk implies the 17-bit current value of the location counter. If the instruction containing an * in the address subfield is assigned to a full word, character position zero is implied. If it is assigned to a partial word (character), character position 0, 1, 2 or 3 may be implied (EQU, C pseudo instruction only).

The code element, **, results in a 17-bit value of all ones.

LACH	r	instruction
LACH	**	special address
LACH	77777 position 3	result

RELATIVE ADDRESS

A symbol in an r or s subfield instruction with a C modifier must be defined elsewhere as a location symbol. The 17-bit value assigned to that location symbol is assigned to this symbol also.

LACH	r	instruction
LACH	ABLE	relative address (ABLE is 000000000000001 10_2 or 00001_8, character position 2)
LACH	00001_8 position 2	result

If the symbol were previously assigned a 15-bit value, 2 zero bits are added to the right resulting in a 17-bit value which is assigned to this symbol.

Coding

```
ABLE          BSS       ARRAY1
               .
               .
               .
              LACH      ABLE
               .
```

The value assigned to ABLE is 00100_8. The symbol ABLE in the address subfield of the LACH instruction is converted to 00100_8, character position 0. Character position 0 of that word is referenced by 00400_8.

LACH	r	instruction
LACH	*	special address (current 17-bit value of location counter is 00000000100000000_2)
LACH	00100_8 position 0	result

An expression in an r or s subfield results in the addition or subtraction of two 17-bit values. In the following examples, ABLE is assigned the value 01234_8, position 1; the 17-bit current value of the location counter is 00014_8, position 1.

Coding			Result (value of A)	Position
A	EQU, C	ABLE+12	01237_8	1
A	EQU, C	ABLE=12	01231_8	1
A	EQU, C	ABLE-*	01220_8	0
A	EQU, C	ABLE+12B	01235_8	3
A	EQU, C	ABLE-12B	01231_8	3
A	EQU, C	*+12B	00016_8	3

WORD TO CHARACTER ADDRESS CONVERSION

A pseudo or machine instruction which requires a character address may contain either a word or character address. A word address is converted to a character address according to the following formula:

word address times 4 = character address

00123_8 times 4 = 000514_8

```
ABLE    LDA     ARRAY1
         .
         .
         .
        LACH    ABLE
```

The load instruction is assigned to location 00100_8. The symbolic address ABLE of the LACH instruction is converted to a character address, 00400_8, character position 0.

CHARACTER TO WORD ADDRESS CONVERSION

A pseudo or machine instruction which requires a word address may contain either a word or character address. A character address is converted to a word address by the assembler according to the following formula:

octal character address ÷ 4 = octal word address

$000514_8 \div 4 = 00123_8$

```
ABLE     BSS, C     10
          .
          .
          .
         LDA        ABLE
```

The ABLE address is 00100_8, character position 0. The symbolic address ABLE of the LDA instruction is converted to word address 00100_8. If the original character address contains ones in the last 2 bit positions before conversion, a T error will be printed on the output listing.

OTHER ADDRESS REPRESENTATION

Other subfields (b, v, ch, x, i, c and 1) may be represented by legal address coding.

INDEX REGISTER

An index register designation is formed for the numbers 1, 2 or 3, a double asterisk, an expression, or a symbol equated to a numeral by the EQU instruction in the b subfield.

```
LDA    m, b              typical instruction
```

In the following examples, ABLE is assigned value 00100_8 and SYM8 is equated elsewhere in the program to number 1.

Coding	Result (octal)		
LDA ABLE, 1	20	1	00100
LDA ABLE, SYM8	20	1	00100
LDA ABLE, **	20	3	00100
LDA ABLE, SYM8+1	20	2	00100

The b subfield may specify indexing or direct usage of the index register; in either case, evaluation must result in a value of 1, 2, or 3.

TIA b typical instruction

In the following example, B1 is equated to value 1 elsewhere in the program.

Coding		Result (octal)		
TIA	B1	53	1	0 - - - -

REGISTER FILE

A location in the register file is formed for a code element in the v subfield. Any coding element resulting in 00_8 through 77_8 may appear in the subfield.

TMA v typical instruction

In the following examples, ABLE is equated to 0011_8 elsewhere in the program.

Coding		Result (octal)				
TMA	ABLE	53	0	2	. . .	11
TMA	77B	53	0	2	. . .	77
TMA	**	53	0	2	. . .	77
TMA	ABLE+22B	53	0	2	. . .	33

CHANNEL NUMBER

A channel number (Input/Output) is formed for a code element in the ch subfield. The ch subfield may contain one number, 0 through 7, or any legal coding element which results in 0 through 7.

INAC ch typical instruction

In the following examples, CHAN2 is equated to the value 2 elsewhere in the program.

Coding			Results (octal)
INAC	CHAN2	word 1	73
		2	2
		3	
INAC	7B	word 1	73
		2	7
		3	
INAC	**	word 1	73
		2	7
		3	

Coding			Results (octal)	
INAC	CHAN2+2B	word 1	73	
		2	4	
		3		

FUNCTIONAL CODE OR LOGICAL MASK

A function code or logical mask is formed for an element of code in the x subfield. The resultant value must be equivalent to a 12-bit number.

CON x, ch typical instruction

In the following examples, LOGMSK is equated to 0111_8 elsewhere in the program.

Coding		Results (octal)			
CON	LOGMSK, 2	77	0	2	0111
CON	LOGMSK+22, 2	77	0	2	0133
CON	22B, 2	77	0	2	0022
CON	**, 2	77	0	2	7777

INTERVAL

An interval of 1 to 8 is formed for an element of code in the i subfield which results in an octal value 0 to 7. A code element of 8 results in octal value 0 in the machine instruction.

MEQ m, i typical instruction

In the following examples, INTRVL is equated to 1 elsewhere in the program; ABLE to 00100_8.

Coding		Results (octal)		
MEQ	ABLE, INTRVL	06	1	00100
MEQ	ABLE, INTRVL+1	06	2	00100
MEQ	ABLE, 2	06	2	00100
MEQ	ABLE, 8	06	0	00100
MEQ	ABLE, **	06	7	00100

CHARACTER

The 6-bit character to be searched for is formed for an element of code in the c subfield.

SRCE c, r, s typical instruction

In the following examples, A is defined elsewhere in the program as the BCD character A or octal value 21; ABLE and BAKER are defined as 00200 and 00100.

Coding		Result (octal)	
SRCE A,ABLE,BAKER	word 1	71	00200
	2	21	00100
	3		
SRCE 21B,ABLE,BAKER	word 1	71	00200
	2	21	00100
	3		
SRCE A+21B,ABLE,BAKER	word 1	71	00200
	2	42	00100
	3		

LENGTH The length of a character field, 1 to 128, to be moved in placed in the ℓ subfield. A field length coded as 128 is interpreted as zero, which directs the computer to move 128 characters.

MOVE ℓ,r,s   typical instruction

In the following examples, ABLE is equated to 100_8 elsewhere in the program; BAKER to 00200_8.

Coding		Results (octal)
MOVE,ABLE,BAKER,BAKER+100B	word 1	72000300
	2	20000200
	3	
MOVE 128,BAKER,BAKER+128	word 1	72000400
	2	00000200
	3	
MOVE 27B,BAKER,BAKER+27B	word 1	72000227
	2	1340020
	3	
MOVE **,BAKER,BAKER+100B	word 1	72000300
	2	77400200
	3	

MACHINE INSTRUCTIONS

OPERATION FIELD		ADDRESS FIELD	INSTRUCTION
00	HLT	m	Unconditional stop; read next instruction from location m
	SJ1	m	Jump if key 1 is set
	SJ2	m	Jump if key 2 is set
	SJ3	m	Jump if key 3 is set
	SJ4	m	Jump if key 4 is set
	SJ5	m	Jump if key 5 is set
	SJ6	m	Jump if key 6 is set
	RTJ	m	Return jump
01	UJP, I	m, b	Unconditional jump
02	IJI	m, b	Index jump; increment index
	IJD	m, b	Index jump; decrement index
03	AZJ, EQ	m	jump if (A) = 0
	NE		jump if (A) $\neq$ 0
	GE		Compare A with zero; jump if (A) $\geq$ 0
	LT		jump if (A) < 0
	AQJ, EQ	m	jump if (A) = (Q)
	NE		jump if (A) $\neq$ (Q)
	GE		Compare A with Q; jump if (A) $\geq$ (Q)
	LT		jump if (A) < (Q)
04	ASE, S	y	Skip next instruction, if (A) = y
	QSE, S	y	Skip next instruction, if (Q) = y
	ISE	y, b	Skip next instruction, if (B^b) = y
05	ASG, S	y	Skip next instruction, if (A) $\geq$ y
	QSG, S	y	Skip next instruction, if (Q) $\geq$ y
	ISG	y, b	Skip next instruction, if (B^b) $\geq$ y
06	MEQ	m, i	Masked equality search
07	MTH	m, i	Masked threshold search
10	ISI	y, b	Index skip; increment index
	ISD	y, b	Index skip; decrement index
	SSH	m	Storage shift
11	ECHA, S	z	Enter A with 17-bit character address
12	SHA	y, b	Shift A
	SHQ	y, b	Shift Q
13	SHAQ	y, b	Shift AQ
	SCAQ	y, b	Scale AQ
14	ENA, S	y	Enter A
	ENI	y, b	Enter index
	ENQ, S	y	Enter Q
15	INA, S	y	Increase A
	INI	y, b	Increase index
	INQ, S	y	Increase Q
16	XOA, S	y	Exclusive OR y and (A)
	XOQ, S	y	Exclusive OR y and (Q)
	XOI	y, b	Exclusive OR y and (B^b)
17	ANA, S	y	Logical product (AND) of y and (A)
	ANQ, S	y	Logical product (AND) of y and (Q)
	ANI	y, b	Logical product (AND) of y and (B^b)

MACHINE INSTRUCTIONS (cont'd)

OPERATION FIELD		ADDRESS FIELD	INSTRUCTION
20	LDA,I	m,b	Load A
21	LDQ,I	m,b	Load Q
22	LACH	r, 1	Load A character
23	LQCH	r, 2	Load Q character
24	LCA,I	m,b	Load A complement
25	LDAQ,I	m,b	Load AQ (double precision)
26	LCAQ,I	m,b	Load AQ complement (double precision)
27	LDL, I	m,b	Load logical
30	ADA,I	m,b	Add to A
31	SBA,I	m,b	Subtract from A
32	ADAQ,I	m,b	Add to AQ
33	SBAQ,I	m,b	Subtract from AQ
34	RAD,I	m,b	Replace add
35	SSA,I	m,b	Selectively set A
36	SCA,I	m,b	Selectively complement A
37	LPA,I	m,b	Logical product with A
40	STA,I	m,b	Store A
41	STQ,I	m,b	Store Q
42	SACH	r, 2	Store character from A
43	SQCH	r, 1	Store character from Q
44	SWA,I	m,b	Store 15-bit word address from A
45	STAQ,I	m,b	Store AQ
46	SCHA,I	m,b	Store 17-bit character address from A
47	STI,I	m,b	Store index
50	MUA,I	m,b	Multiply A
51	DVA,I	m,b	Divide AQ (48 by 24)
52	CPR,I	m,b	Within limits test
53	TIA	b	Transmit (B^b) to A
	TAI	b	Transmit (A) to B^b
	TMA	v	Transmit (high speed memory) to A
	TAM	v	Transmit (A) to high speed memory
	TMQ	v	Transmit (high speed memory) to Q
	TQM	v	Transmit (Q) to high speed memory
	TMI	v,b	Transmit (high speed memory) to B^b
	TIM	v,b	Transmit (B^b) to high speed memory
	AQA		Transmit (A) + (Q) to A
	AIA	b	Transmit (A) + (B^b) to A
	IAI	b	Transmit (B^b) + (A) to B^b
54	LDI,I	m,b	Load index
55	EUA		Transmit (E upper) to A
	ELQ		Transmit (E lower) to Q
	AEU		Transmit (A) to E upper
	QEL		Transmit (Q) to E lower
	EAQ		Transmit (E upper) to A and (E lower) to Q
	AQE		Transmit (AQ) to E

MACHINE INSTRUCTIONS (cont'd)

OPERATION FIELD		ADDRESS FIELD	INSTRUCTION
56	MUAQ,I	m,b	Multiply AQE (96 by 48)
57	DVAQ, I	m,b	Divide AQE (48 by 48)
60	FAD,I	m,b	Floating add to AQ
61	FSB,I	m,b	Floating subtract from AQ
62	FMU,I	m,b	Floating multiply AQ
63	FDV,I	m,b	Floating divide AQ
64	LDE	r,1	Load E
65	STE	r,2	Store E
66	ADE	r,3	Add to E
67	SBE	r,3	Subtract from E
70	SFE	y,b	Shift E
	EZJ,EQ	m	Compare E with zero; jump if E = 0
	LT		Compare E with zero; jump if E < 0
	EOJ	m	Jump to m on E overflow
	SET	y	Set D to value of y
71	SRCE, INT	c,r,s	Search character equality
	SRCN, INT	c,r,s	Search character inequality
72	MOVE, INT	l,r,s	Move y characters from m_1 to m_2
73	INPC, INT, B, H	ch,r,s	Input character block to memory
	INAC, INT	ch	Input character to A
74	INPW, INT, B, N	ch,m,n	Input word block to memory
	INAW, INT	ch	Input word to A
75	OUTC, INT, B, H	ch,r,s	Output character block from memory
	OTAC, INT	ch	Output character from A
76	OUTW, INT, B, N	ch,m,n	Output word block from memory
	OTAW, INT	ch	Output word from A
77.0	CON	x,ch	Connect
77.1	SEL	x,ch	Select
77.20	COPY	x,ch x = 0	Copy status
77.2	EXS	x,ch x ≠ 0	External sense
77.3	INS	x,ch	Internal sense
77.4	INTS	x,ch	Interrupt sense
77.50	INCL	x	Interrupt clear
77.51	IOCL	x	I/O clear
77.52	SSIM	x	Selective set interrupt mask
77.53	SCIM	x	Selective clear interrupt mask
77.6	PAUS	x	Pause
77.70	SLS		Selective stop
77.71	SFPF		Set floating point fault
77.72	SBCD		Set BCD fault
77.73	DINT		Disable interrupt control
77.74	EINT		Enable interrupt control
77.75	CTI		Console typewriter in
77.76	CTO		Console typewriter out
77.77	UCS		Unconditional stop

PROGRAM ASSEMBLY 2

ASSEMBLY INPUT

Assembly input data is on cards or card images containing octal, mnemonic, or pseudo instructions. A subprogram may begin with an IDENT instruction card and terminate with an END card. Input decks for subsequent use with a monitor system on larger equipment configurations also require IDENT and END cards.

IDENT m

IDENT must be the first instruction of each subprogram; if it appears again anywhere else before an END instruction, it will be flagged with an O error and ignored.

The address field must contain a legal symbol. This symbol is the name of the subprogram, and will appear with the length of the subprogram in the first card (IDC) of the binary object deck. A symbol in the location field is not assigned a value and should not be referred to in subsequent program instructions.

END m

END, which signals termination of a subprogram, produces a TRA card as the last card in the binary object deck. A symbol in the address field will appear as the symbolic transfer address on the TRA card. If a symbol is in the location field, it is not assigned a value and should not be referred to in subsequent program instructions.

PSEUDO INSTRUCTIONS

The following pseudo instructions assign locations, define data, reserve storage, simulate machine instructions with octal codes.

ORGR m

ORGR designates the value in the address field as the beginning location for subsequent instructions. A symbol in the address field must be previously defined elsewhere in the program as a location symbol.

Example:

```
            ORGR     00100
START       LDA      ABLE
            LDA      BAKER
             .
             .
             .
```

In the above example, START is assigned value 00100 and START+1 is assigned to 00101.

DATA DEFINITION

Constant data is assembled into a program with data definition pseudo instructions. Binary coded decimal, octal, or decimal constants may be inserted into machine words with OCT, BCD, DEC or DECD. Character positions (6 bits of a machine word) may be filled with constants by the BCD, C pseudo instruction.

OCT m

OCT stores an octal constant into a machine word. Although not required, a constant may be preceded by a plus or minus sign; an unsigned constant is assumed positive. A maximum of 8 octal digits may be contained in an octal constant. If there are less than 8 digits, the constant is right justified in the word. A location symbol defines a 15-bit word address:

Examples:

```
OCT    77777777
OCT    +1
OCT    -57
OCT    2040
```

	results
word 1	77777777
2	00000001
3	77777720
4	00002040

DEC d

DEC converts a signed or unsigned fixed point decimal constant to binary and stores it in a machine word.

<u>Decimal Integer</u> is a sign followed by 1 to 7 decimal digits. If the sign is omitted, the integer is assumed positive. The decimal integer may be followed by a decimal or a binary scaling factor or both in either order.

Decimal Scaling Factor consists of D±d. D indicates decimal scaling; d may not exceed two decimal digits. The largest practical decimal scaling factor is 14.

Binary Scaling Factor consists of B±b. B indicates binary scaling; b consists of up to two decimal digits not greater in magnitude than 23.

The magnitude of the constant after scaling must be less than 2^{23}. The conversion is performed in three steps:

1. The decimal integer is converted to a binary integer which must be less than or equal to $2^{23}-1$.
2. The binary integer is multiplied or divided by 10^d (d is decimal scaling factor). The magnitude of the result must be less than 2^{47}. If the decimal scaling factor is negative, a 47-bit fraction or mixed fraction is formed.
3. The result is shifted the number of bits specified by the binary scaling factor. A negative factor produces a right shift; a positive scale factor a left shift. If non-zero bits are lost from the high order 24 bits of the result, an error is flagged. Low order bits of the intermediate result may be lost and not flagged.

Examples:

1	decimal integer
+2	decimal integer
-38	decimal integer
1D5	decimal integer, decimal scale factor
73D-2	decimal integer, decimal scale factor
-6D+1B4	decimal integer, decimal and binary scale factors
200B-7	decimal integer, binary scale factor
36B+2D1	decimal integer, binary and decimal scale factors

DECD d

DECD converts a signed or unsigned double precision decimal constant to binary and stores it in two consecutive machine words. Fixed point or floating point constants may be specified.

Floating point constant may be a signed or unsigned decimal integer up to 14 digits. A decimal point, which may appear anywhere within the integer, identifies it as a floating point constant. A decimal scale factor is permitted; the result after scaling must not exceed the capacity of the hardware (approximately $10^{\pm 308}$). A binary scale factor is not permitted.

Fixed point constant format is identical to that of the DEC single precision constants; however, magnitudes may be larger. Up to 14 decimal digits may be specified, expressing a value of not more than 2^{47}. Decimal and binary scale factors may be used as in the DEC pseudo operation. Low order bits are not lost; the signed 48-bit binary result is stored into two consecutive computer words.

BCD $n, c_1 c_2 \ldots c_{4n}$

BCD converts keypunch characters to standard BCD code and stores them in consecutive 24-bit machine words. The address field consists of a decimal number n, followed by a comma and the characters, including blanks; the character string ends before column 73.

The result is n computer words each containing four BCD characters. Anything after 4n characters is treated as remarks. If the value of n exceeds the number of punched characters on the card, blanks are filled in for the excess. The location field may be blank or contain a symbol which is converted to a 15-bit address.

Example:

BCD 2,ABCD

Octal contents of machine words:

word 1	21	22	23	24
2	blanks			

BCD 12,ABCDEFGHIJKLMNOPQRSTUVWXYZ=-+ +0 .)- -0 $ *(blank) /,(12345678

Note 1: The characters in the above line comprise the complete BCD character set. Normally, the code would be contained on one line with no spaces between the characters except for specified blanks.

Note 2: If this word instruction follows any instruction which left a partial word, the balance of the partial word is unused.

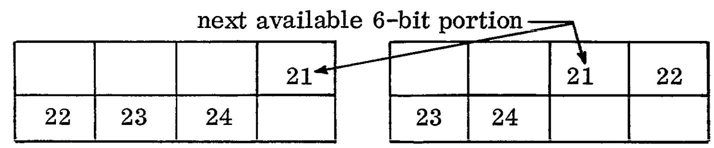
word				
1	A 21	B 22	C 23	D 24
2	E 25	F 26	G 27	H 30
3	I 31	J 41	K 42	L 43
4	M 44	N 45	O 46	P 47
5	Q 50	R 51	S 62	T 63
6	U 64	V 65	W 66	X 67
7	Y 70	Z 71	= 13	- 14
8	+ 20	+0 32	. 33	) 34
9	- 40	-0 52	$ 53	* 54
10	b 60	/ 61	, 73	(74
11	1 01	2 02	3 03	4 04
12	5 05	6 06	7 07	8 10

BCD,C $n, c_1 c_2 \ldots c_n$

BCD, C converts keypunch characters to standard BCD code and stores them in consecutive 6-bit portions of consecutive 24-bit machine words. This instruction is similar to BCD except that a character is assigned to the next available 6-bit portion of a machine word. The address field contains n, followed by a comma, and n standard keypunch characters; the character string ends before column 73. If the value of n exceeds the number of punched characters on the card, blanks are filled in for the excess. The location field may be blank or contain a symbol which is converted to a 17-bit character address.

Example:

BCD, C   4, ABCD



If this character instruction follows any instruction which left a partial word, the filling of constants begins at the next unassigned character position in the partial word.

Example:

Intersperse constants with machine instructions.

Coding			Results (octal and mnemonic)
	.		
CON1	BCD	4, ABCDABCDABCDABCD	21 22 23 24
	LDA	2200B	21 22 23 24
	UJP	13B	21 22 23 24
CON2	BCD, C	1, A	21 22 23 24
CON3	BCD, C	2, BC	LDA 02200
CON4	BCD, C	5, DEFGH	UJP 00013
CON5	BCD, C	1, A	21 22 23 24
	LDA	1500B	25 26 27 30
	.		21 00 00 00
	.		LDA 01500
	.		

AREA RESERVATION

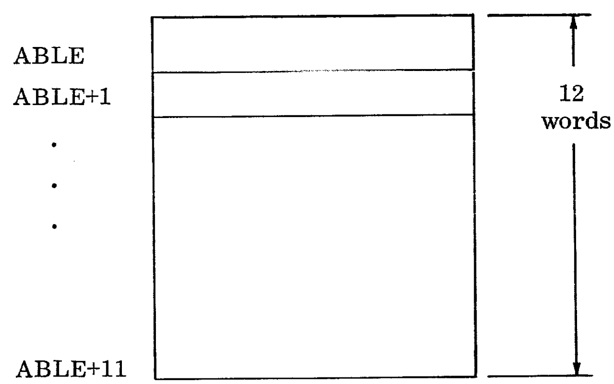
The following pseudo instructions, BSS and BSS, C, reserve a block of data storage as words or as characters. The resultant value of the address field must be positive and non-relative.

BSS m

BSS reserves a block of consecutive, 24-bit machine words specified by m. The address field contains any element of code which results in a positive integer. If a symbol is used, it must be defined previously in the program. If m is zero, the next storage assignment is forced to the beginning of a new word. Word locations within the block may be established by address arithmetic or indexing. The location field may be blank or contain a symbol which defines a 15-bit word address representing the first location of the block.

Example: Reserve a block of 12 words.

```
ABLE BSS 12
```



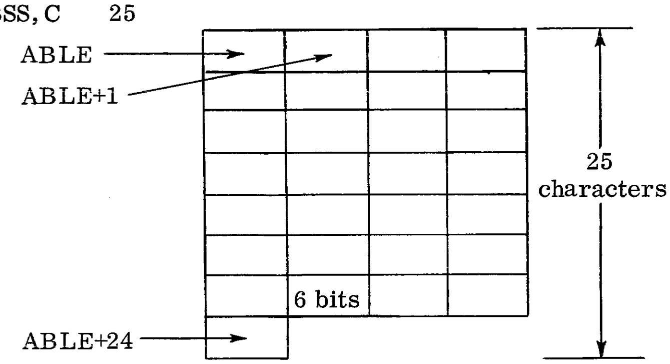
If the instruction preceding ABLE were assigned to location 777_8, the 12-word block would be assigned to locations 1000_8 through 1013_8. The second word within the block could be reached by the coding element ABLE+1 or by indexing.

BSS,C m

BSS, C reserves a block of character locations, 6-bit word portions; the address field contains any element of code which results in a positive integer. If a symbol is used, it must be defined previously in the program. m specifies the number of consecutive character locations (4 m words) to be reserved within the block. One location symbol may be assigned to the block; it defines a 17-bit character address which refers to the first character position of the block. Character locations within the block may be reached by address arithmetic or indexing.

Example: Reserve a block of 25 characters.

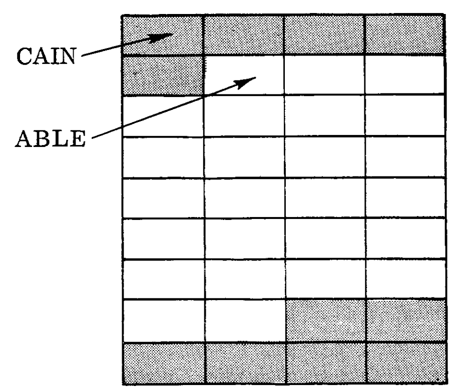
```
ABLE   BSS,C   25
```



If the instruction preceding ABLE were assigned to word location 777_8 (a 15-bit address), the 25-character block would be assigned to word locations 1000_8 to 1006_8 which correspond to character locations 4000_8 to 4030_8. The second character could be reached by the coding element, ABLE+1, and the last character could be reached by the code element, ABLE+24. If the instruction preceding ABLE were terminated before the last character location within a word (a 17-bit address), the 25-character block would be assigned to the next available character location (17-bit address).

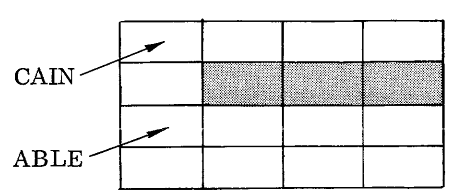
Example: Reserve a 25-character block immediately following a 5-character block.

```
CAIN   BSS,C    5
ABLE   BSS,C   25
```



Reserve an 8-character block in the character position 0 of a word following a 5-character block.

```
CAIN   BSS,C   5
       BSS     0
ABLE   BSS,C   8
```



EQUATE

The pseudo instructions EQU and EQU, C equate symbols to other symbols or to values.

EQU m

EQU equates a location symbol to an address field symbol or value. Address field symbols must be previously defined (used as location symbols earlier in the program). The location symbol defines a 15-bit word address.

Example: Equate a symbol to a previously defined symbol.

```
ABLE     BSS     10
          .
          .
          .
TIM      EQU     ABLE+4
          .
          .
          .
         LDA     TIM
```

If ABLE were assigned to 01000_8, TIM would be assigned 01004_8. If an instruction subsequent to EQU addresses TIM, 01004_8 will be assigned.

Equate a symbol to a value.

```
 TOT     EQU     7B
ELDER    EQU     99
         RAD     TOT
         ADA     ELDER
```

The symbol TOT is assigned the value 00007_8, any place subsequent to the EQU instruction, TOT will be assigned 00007_8. The symbol ELDER is assigned the value 00143_8. Any subsequent use of ELDER in the address field results in the value 00143_8 being assigned.

EQU,C r

EQU, C equates a location symbol to an address field symbol or value. Address field symbols must be previously defined (used as location symbols earlier in the program). The location symbol defines a 17-bit character address.

Examples:

Equate a symbol to a character address.

```
ARRAY    BSS,C    10
           .
           .
           .
CHAR5    EQU,C    ARRAY+5
           .
           .
           .
         LACH     CHAR5
```

If ARRAY were assigned to 01000_8, position 0, CHAR5 would be assigned to 01001_8, position 1. If an instruction subsequent to EQU, C addresses CHAR5, 01001_8, position 1 will be assigned.

Equate a symbol to a word address.

```
ARRAY    BSS      10
           .
           .
           .
CHAR5    EQU,C    ARRAY+5
           .
           .
           .
         LACH     CHAR5
```

If ARRAY were assigned to 01000_8, CHAR5 would be assigned to 01005_8, position 0. If an instruction subsequent to EQU, C addresses CHAR5, 01005_8, position 0 will be assigned.

Equate a symbol to a value.

```
ABLE     EQU,C    777B
BAKER    EQU,C    009
         LACH     ABLE
         LACH     BAKER
```

The symbol ABLE is assigned 00177, position 3. The symbol BAKER is assigned 00002, position 1.

ASSEMBLY OUTPUT

Output from the assembly consists of two types:

Output listing

Binary output for subsequent loading and execution of the assembled program

Binary output is a machine language program on cards or card images in relocatable binary format that may be loaded into any portion of storage at run time.

OUTPUT LISTING

The listing contains error codes, machine locations, the assembled contents of the machine location number, and the input coded machine, octal or pseudo instructions (location, operation, address and comments fields).

ERROR CODES

The following error codes may appear in the leftmost columns of the assembly listing:

A An illegal character or coding element in the address field.

D The same symbol is used in more than one location field term. Only the first symbol is recognized.

F Symbol table is full. No more location field symbols will be recognized.

L A symbol appears in the location field when not permitted, a symbol is missing in the location field when one is required, or an illegal location symbol appears.

M A modifier appears in the operation field when not permitted, a modifier is missing in the operation field when one is required, or an illegal modifier appears in the operation field.

O Illegal operation code. Zeros are substituted for the operation code.

U Undefined symbol. The assembler assigns the symbol to a region following the last program entry.

T A character symbol was used in an address subfield requiring a word symbol. Significant bits are lost.

LOCATION FIELD ERROR

The EQU and EQU, C pseudo instructions must have a symbol in the location field, otherwise the instruction is assigned an error code L. The following pseudo instructions may have a location symbol; an error code L signifies an illegal symbol:

BSS	BCD, C
BSS, C	OCT
ORGR	DEC
BCD	DECD

The program identification pseudo instructions (IDENT and END), the assembly listing control pseudo instructions (EJECT, SPACE, LIST, NOLIST and REM), and the pseudo instruction ORGR may have a valid symbol in the location field. The assembler does not define a symbol placed in the location field of these instructions because they do not use storage space; that symbol is not assigned a value. Subsequent instructions which refer to the symbol will be flagged with a U error (undefined symbol). A symbol placed in the location field of one of these instructions may be in the location field of other instructions and a D error (doubly defined symbol) will not occur.

ADDED LISTINGS

The assembler produces a list of undefined symbols, doubly defined symbols, the length of the subprogram, and a count of the output lines which contain error flags, and an indication of the presence of instructions in the code which may be trapped.

LISTING FORMAT

Listable output format is as follows:

Column	
1	carriage control
2-9	error codes
10	blank
11-16	octal location
17	blank
18-19	octal contents of operation field
20	blank
21	octal contents of j field
22	blank
23-27	octal contents of address field
28	blank
29-108	source card

LISTING CONTROL

The assembly listing may be controlled with EJECT, SPACE, NOLIST, LIST and REM pseudo instructions. If a symbol appears in the location field of one of these instructions, that location symbol is not assigned a value and should not be referred to in subsequent program instructions.

EJECT moves the paper to the top of the next page. The next instruction will be printed as the top line on the next page.

SPACE spaces the output listing the number of lines specified in the address field. If the spacing would cause an overflow at the bottom of the page, the page is ejected to the top of the next page only.

NOLIST suppresses the printing of assembly lines until a LIST pseudo instruction is encountered. However, lines with error codes will be printed and the NOLIST line will be printed.

LIST resumes printing. LIST is recognized by the assembler only if a NOLIST has been encountered previously.

REM produces a printed line containing remarks only. All columns, except 9 to 13, from the assembly coding sheet are printed as remarks.

RELOCATABLE BINARY CARD OUTPUT

The relocatable binary card deck produced by the assembler may be used by the relocatable loader or by a simple loader. The deck contains elements that enable the loader to relocate coded information. The deck consists of the following card types which are usually produced in the order listed:

IDC Program Identification Card

specifies the program and its length.

EPT Program Entry Point Card

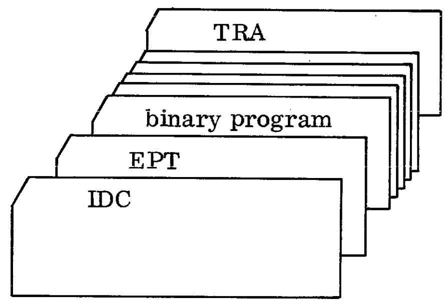
contains the entry point, if a symbol appeared in the END card.

RIF Relocatable Information Card

contains program information to be loaded into storage.

TRA Program Transfer Card

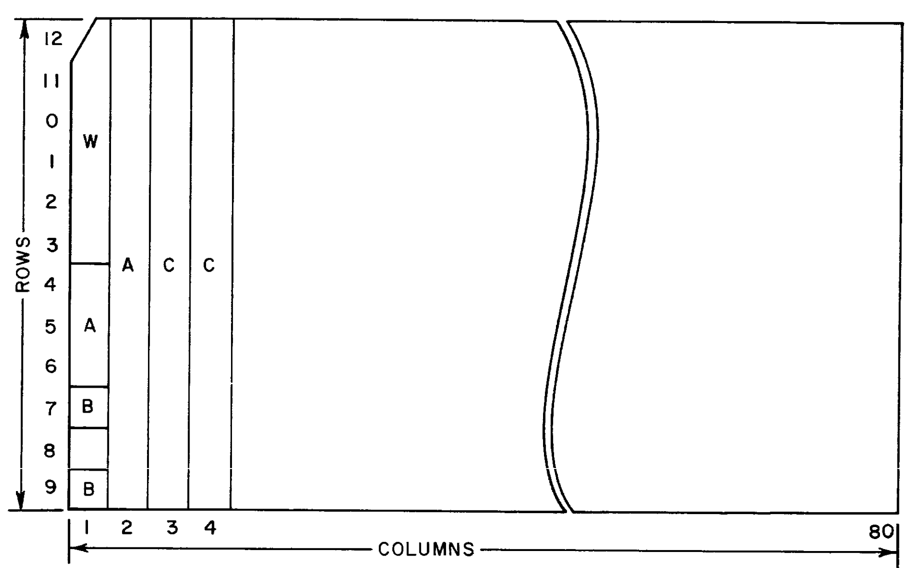
indicates the end of the program and contains the transfer point.



BINARY CARD DESCRIPTION

All binary cards contain a 7 and 9 punch in column 1. The first two columns identify the type of card and provide a means of checking its contents.

Mnemonic	Rows	Card Column	Computer Word Bit Position	Purpose
w	12,11,0-3	1	23-18	Word Count
a	4,5,6	1	17-15	Address, sequence number, or program length
	12,11,0-9	2	11-0	
b	7,9	1	14 and 12	7-9 punch
c	12,11,0-9	3	23-12	24-bit checksum
	12,11,0-9	4	11-0	
i	8	1	1	Ignore checksum



IDC Identifies the subprogram which follows and provides subprogram name.

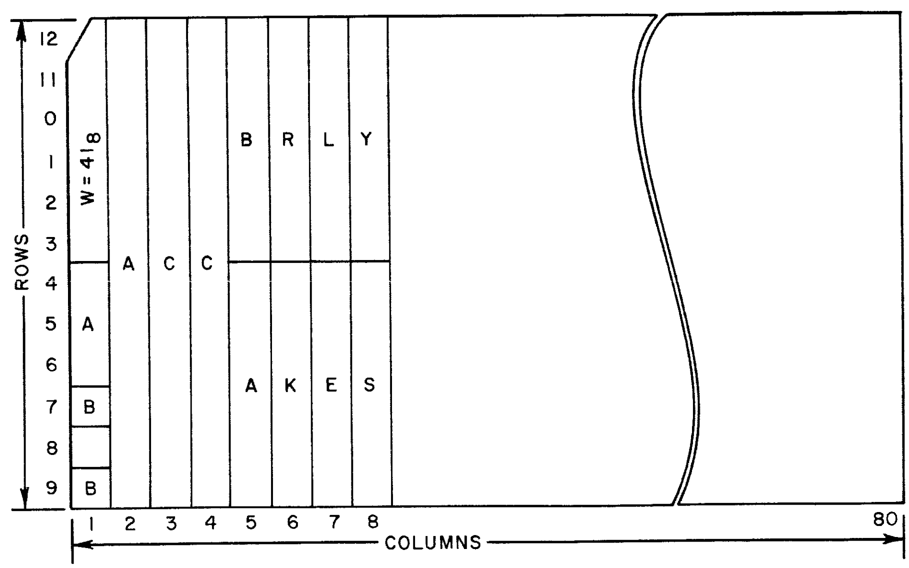
Card Content:

Columns	Computer Word	Use
1-2	1	Standard card type identification
3-4	2	Checksum
5-8	3-4	Program name in BCD
9-80	5-40	Unused

Word Content:

1	w = 41_8, a = Subprogram length in words
2	c = Checksum
3-4	Program name in BCD †
5-40	Unused

This card contains program name, BARKLEYS



† The name is 8 characters or less, formed according to the rules for symbols. Words 3 and 4 are used for the name; trailing blanks are added.

EPT The program entry point card contains the symbolic entry point name and its program address (relative). An EPT card is produced if a symbol appears in the END card.

Card Content:

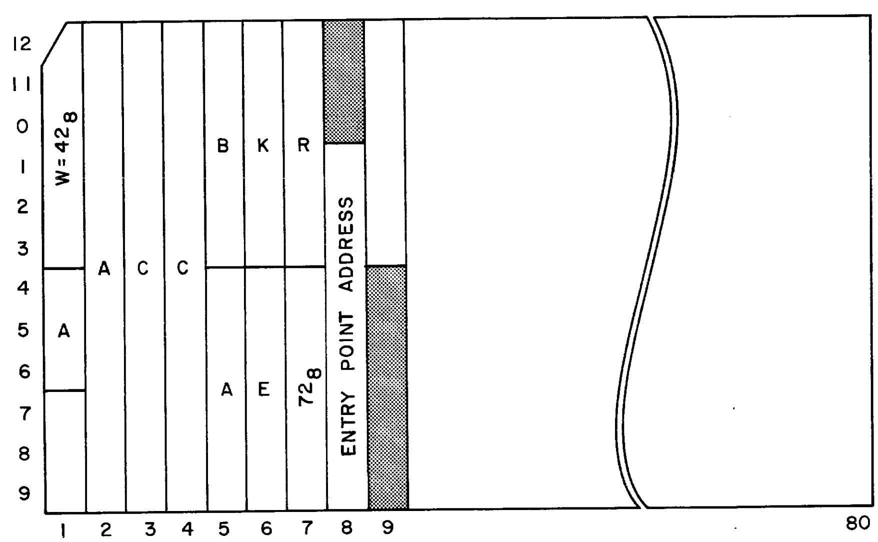
Columns	Computer Word	Use
1-2	1	Standard card type identification
3-4	2	Checksum
5-10	3-5	Entry point (transfer point) name and its program address
11-80	6-40	Unused

Word Content:

1	w = 42_8, a = 1
2	c = Checksum
3-5	Entry point name and location
6-40	Unused

A 1 to 6 character name is followed by a record mark character (internal code 72_8), and 18 bits of which the rightmost 15 specify the entry point address.

This card contains the entry point, BAKER.



RIF The relocatable information cards contain the binary representation of the assembled program.

Card Content:

Columns	Computer Word	Use
1-2	1	Standard card type identification
3-4	2	Checksum
5-16	3-8	Relocation bytes
17-80	9-40	Program information

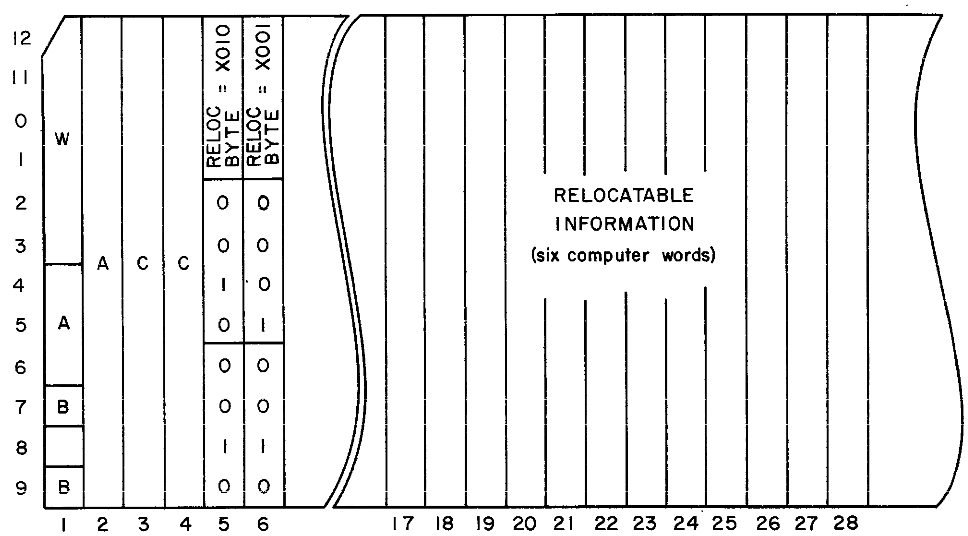
Word Content:

1	w is the word count, 1 to 40_8 a is the load address of the first information word on the card. Succeeding words are loaded into sequential locations.
2	24-bit checksum
3-8	may contain up to 33 relocation bytes
9-40	contain the relocatable binary information to be loaded

A relocation byte, 4 bits/byte, specifies the type of relocation to be applied to the load address and the address field of each word on the card. The first bit of each byte indicates whether the relocation is applied to a 15-bit (word) address or a 17-bit (character) address.

The following is a list of relocation codes:

X001	Absolute reference (no relocation)
X010	Program increment
X101	Program decrement
X000	Relocation error



This card contains six words of relocatable binary information.

TRA The transfer card closes the binary program deck; it is produced by the appearance of an END card in the assembly input deck.

Card Content:

Columns	Computer Words	Use
1-4	1 and 2	Standard card type identification
5-10	3 through 5	Entry point name of the starting address of the program, in Hollerith. When there is no transfer name, columns 5 through 10 are blank.

Word Content:

1 w = 44_8, a is the transfer address

2 c = Checksum; formed by computing the sum of all binary card checksums. Computing is done modulo $2^{24}-1$.

PAPER TAPE OUTPUT

The relocatable binary card format may be produced on paper tape, and the preceding format is retained except as follows:

Each card column is represented by 2 frames of paper tape using tracks 1 through 6. The first frame represents rows 12 through 3, the second frame represents rows 4 through 9.

A seventh level punch appears in the first frame of each card image.

The number of frames punched for each card image is variable, consisting of the control information and as many frames as needed to contain the data.

EXECUTIVE PROCESSOR 3

BASIC assembler consists of the assembly processor and a set of input/output driver subroutines for physical units. The same input/output format is used by the processor regardless of the peripheral equipment used. For example, a code line punched on paper tape in standard Flex code is edited and recoded as an 80-column BCD card image.

INPUT/OUTPUT TO BASIC ASSEMBLER

A version of the BASIC assembler may be requested to include the drivers for any combination of input and output units.

Input	Listable Output	Binary Output
Magnetic Tape	Magnetic Tape	Magnetic Tape
Paper Tape Reader	Paper Tape Punch	Paper Tape Punch
Card Reader	Card Punch	Card Punch
	Line Printer	
	Typewriter	

The four types of I/O driver subroutines provided by BASIC assembler are listed below:

I/O Driver Subroutine	Magnetic Tape	Card Reader	Card Punch	Paper Tape	Line Printer	Type-writer	Number of Characters Processed
BCD Input	x	x		x			80
BCD Output			x	x		x	80
	x				x		120
Binary Output	x	x		x			80
Binary Input	x	x		x			80

BCD and binary input driver subroutines provide a standard 80 character card image. BCD output driver subroutines accept a standard 120 character print image; binary output driver subroutines accept 80 columns of binary card images. BCD output driver subroutines for card, paper tape and typewriter process the first 80 characters only. The programmer specifies input and output for the BASIC assembler at the halt preceding entry to the System Initializer Routine of the control program.

PARAMETER ENTRIES

Requests are made by entering parameters in the A register.

Parameters in Octal	Physical Unit
0X	Magnetic tape X on channel 0
1X	Magnetic tape X on channel 1
20	Paper tape reader or punch
30	Card reader
40	Card punch
50	Line printer
60	Typewriter (console)
77	No physical unit

X range, 0-7

Examples:

Select magnetic tape 1, channel 0 as input unit, magnetic tape 2, channel 0 as listable output unit, magnetic tape 3, channel 0 as binary output unit.

0 1	0 2	0 3	7 7
I	L	B	O

A Register (octal)

Select a card reader as the input unit, a printer as listable output unit, and a card punch as binary output unit.

3 0	5 0	4 0	7 7
I	L	B	O

I = driver subroutine that reads one source card image (INPUT).

L = driver subroutine that processes listable output.

B = driver subroutine that processes binary output.

O = used by the relocating loader to load the BASIC assembler.

ENTRY TO DRIVER SUBROUTINES

Entries may be made to driver subroutines by a return jump instruction after first setting the A and Q registers and console jump switches. The A register contains the unit identity in bits 5 through 0.

CONSOLE JUMP SWITCHES

The operator may suppress the output list by setting switch one or the binary output by setting switch two.

I/O HALTS

I/O halts may occur during the run of a BASIC assembly program. Either an error is signaled after a reasonable attempt has been made to recover, or action by the operator is requested. For any I/O halt, Q contains zeros, the unit is identified in index register B3 and A contains the halt code.

Halt Code	Meaning
00000040	Illegal function request (undefined function code)
00000041	I/O unit malfunction (parity errors, lost data, compare errors)
00000042	Illegal hardware reject of function request
00000050	Feed failure
00000051	Hopper empty
00000052	Stacker full
00000053	Out of paper tape
00000054	Out of paper
00000060	Reposition the input file

SYSTEM LIBRARY

The BASIC system library is an autoloaded control program and a library of routines in relocatable form. The system library may be recorded on cards, magnetic tape, or paper tape. The Control Program (COP) is composed of card images output by the Prepare Library Program (PLIB). The first card contains a bootstrap loader which occupies the lowest portion of storage and reads in the remainder of the Control Program.

The relocatable library contains driver subroutines for input/output devices, the library preparation program (PLIB), and the BASIC assembly program with related routines in relocatable binary format.

The system library unit is arranged as follows:

low order storage

Bootstrap Loader	Control Program (COP)
Relocatable Loader	
Library Driver Routine	
System Initializer	
Transfer Table	
Memory Table	
Transfer Card (TRA)	Relocatable Library
Driver Subroutines	
EOF	
BASIC Assembler	
PLIB	

high order storage

Items from the system library are positioned in storage in the following order:

low order storage

Interrupt Cells
Transfer Table
Memory Table
Relocatable Loader
Library Driver Routine
System Initializer
BASIC Assembly Program
Driver Subroutines

high order storage

CONTROL PROGRAM

COP accomplishes the following:

- Bootstrap loads a relocatable loader
- Clears interrupt cell table
- Sets up and maintains storage limits
- Sets up I/O driver requirements
- Loads and links the driver routines
- Loads and enters the BASIC assembly program

BOOT

The bootstrap loader is read into low storage. BOOT receives control at location 00000 from the autoload sequence and begins to load the program which follows it on the library unit. Only relocatable binary cards and a single transfer card are loaded.

TABLES

Two tables are maintained by the Control Program, the transfer and the memory table.

Transfer Table records physical unit identification and driver entry point addresses. A unit identification and entry point comprise one word in the table as follows:

Unit (octal code)		Driver Entry Point Address
23 — 18	17 — 15	14 — 0

All entries in the table are initially set the same as location 00013_8 to point to the library unit driver. Table entries at specific locations define the functions:

Location	Function
00013	Library unit driver and identity
00014	Input unit driver and identity
00015	Listable output unit driver and identity
00016	Binary output unit driver and identity
00017	BASIC assembly program unit driver and identity

A 2-digit octal code indicates the physical unit to be driven by the subroutine.

Memory Table indicates the current bounds of storage. Bits 14 through 0 of location 00020_8 contain the address of the first available storage word; bits 14 through 0 of location 00021_8 , the last available storage word.

SYSTEM INITIALIZATION

SIN, the system initializer, is responsible for the following:

- Setting storage limits
- Determining I/O driver subroutine requirements
- Loading driver subroutines and establishing linkage to them in the Transfer Table
- Loading and entering the BASIC assembler routines

A programmed halt at the beginning of the system initialization phase permits the operator to insert I/O unit identification in the A register. I/O unit identities are processed in the A register in this order:

Input	Listable output	Binary output	Main program
23 — 18	17 — 12	11 — 6	5 — 0

The system initializer stores the unit identities into corresponding Transfer Table entries and searches the library for the required drivers. Drivers are recognized from the name on the IDC card:

IOD. X0 — X represents the left most digit of the unit identifier code. For example, a magnetic tape driver is identified by the name, IOD. 00. Though tapes may have codes ranging from 00 through 17, the identity or left most digit is defined as a 0, not as a 0 or 1.

The system relocatable loader loads each driver and the transfer address returned from the loader is entered into the Transfer Table entry for the system unit. If any unit is specified that is the same as the library unit, no driver is sought; also, no driver is sought for the library unit. If a required driver is not found before the driver series ends on the library unit, an error stop occurs. An end-of-file mark terminates the driver routines on the library unit; and when all drivers have been loaded, the library unit is positioned past the end-of-file.

Main Program Loading

The BASIC assembly program is loaded and receives control from SIN, after the I/O drivers have been loaded.

System Initializer Re-entry

SIN may be re-entered through RTJ linkage set up by entry to the BASIC assembler. Upon re-entry to SIN, the A and Q registers are cleared and halt occurs. The operator may enter a quantity into the A register. When the program is re-started, the A register is tested for zero; if it is non-zero a program is loaded from the program unit.

Storage occupied by the previous program, not including I/O drivers, is released. Zero in the A register causes the program just executed to be re-entered. Only SIN Re-entry is kept in storage during execution of the loaded program.

RELOCATABLE LOADER

The relocatable loader loads the object program produced by the BASIC assembly program. The loader may be called by SIN, BASIC assembler, or another program.

CALL PARAMETERS

Loader call parameters are entered into the A register and index register 3. If bits 14-0 are zero, the IDC card has not been read; otherwise, bits 14-0 contain the first word address of the card image in storage. The unit containing the program to be loaded is identified in index register 3 by the units index in the Transfer Table (library unit = 0, input unit = 1, etc.) A number greater than 4 is an error, and a halt occurs.

The loader is called to load the next program in sequence from the unit specified in register 3. If an IDC card has been read, loading continues with the next binary card images. If not, loading begins with the next encountered IDC card; intervening card images are ignored.

CARD PROCESSING

Identification (IDC), relocatable binary (RIF), and transfer (TRA) cards are processed.

IDC Card

The IDC card identifies the beginning of a program deck. The program length contained on the card is used to assign an area of storage for the program, and to form the relocation factors applied during loading. The high address of available storage, minus program length, plus one, is the relocation increment; the complement of the increment is the program relocation decrement.

RIF Card

Relocatable binary information is loaded into sequential locations beginning at the load address plus the program relocation increment. The number of words to be loaded is indicated by the word count. Relocation bytes specify the type of relocation for the word and load address portion of each instruction. The first byte applies to the load address; its value must be 0010_2.

TRA Card

A transfer card signals the end of a program. If there is no address on the card, the first location of the program is the transfer address. If there is an address, the address plus the program relocation increment is the transfer address. Before returning to the calling program, the transfer address is placed in bits 14 - 0 of the A register and the count of errors detected by the loader is placed in the Q register.

ERROR DETECTION

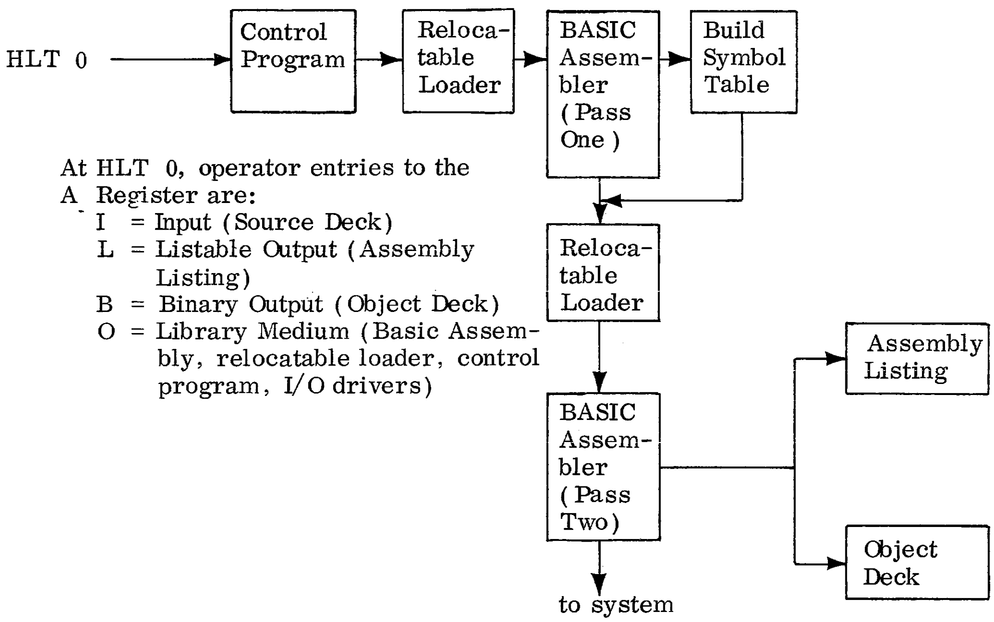
Errors are detected during the several phases of the Control Program and Relocatable Loader. Error conditions discovered by the system result either in an immediate halt or an increment of the error count. Error halt codes are displayed in the A register. The loader error halt is indicated by $0000003X_8$ where X is a digit 0 through 7 further identifying the error.

Halt	Reason	Operator Action
Bootstrap Loader Phase, $0000001X_8$		
00000010_8	No transfer address on program read by BOOT.	Job termination. Punch transfer address.
System Initializer Phase, $0000002X_8$		
00000020_8	Required driver not on tape.	Put required driver on library tape. Restart.
00000021_8	Loader errors.	Display errors count in index register 3
Relocatable Loader Phase, $0000003X_8$		
00000030_8	Memory overflow.	Job termination. Restart passes control to calling program.
00000031_8	Program Checksum from the TRA card does not agree with checksum generated by the loader.	Restart. Program resumed.
00000032_8	Multiple IDC cards.	Job termination. There should be one IDC card.
00000033_8	I/O unit error. Unit number is greater than 4.	Display index register 3 for incorrect unit number. Possible job termination.

Halt	Reason	Operator Action
no halt	Relocation Byte. Load address relocation byte is not 0010_2. Illegal relocation byte given for data address. Error count incremented; processing resumed as if byte were 0010_2.	None
Input/Output, $0000004X_8$		
00000040_8	Input/Output Error. Illegal Function request.	Display index register 3 for unit identity. Job termination or restart after correct function request inserted.
00000041_8	I/O unit malfunction (Parity error, lost data, etc.)	Display index register 3 for unit identity. Job termination.
00000042_8	I/O error. Illegal Hardware reject of function request.	Display index register 3 for unit identity. Job termination.
$0000005X_8$		
00000050_8	I/O error. Feed failure.	Display index register 3 for unit identity. (If accompanied by hopper empty, cards are all read; if not, a feed problem exists.)
00000051_8	I/O error. Hopper empty.	Display index register 3 for unit identity. All cards have been read.
00000052_8	I/O error. Stacker full.	Display index register 3 for unit identity. To prevent a card jam, remove accumulated cards from stacker.
00000053_8	I/O error. Out of Paper Tape.	Display index register 3 for unit identity. Fill paper tape reader.
00000054_8	I/O error. Out of Paper.	Display index register 3 for unit identity. Load paper into printing mechanism.
00000060_8	I/O error. Reposition the input file.	Display index register 3 for unit identity. Position the Input File to the beginning of the file.

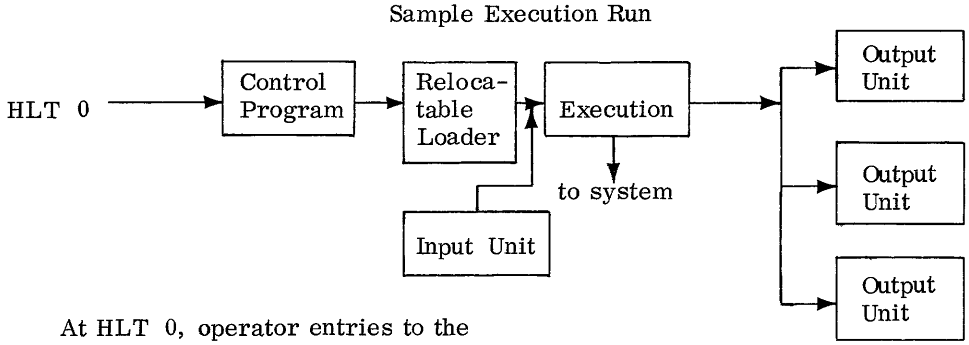
SAMPLE ASSEMBLY RUN — A

Sample Assembly Run



At HLT 0, operator entries to the A Register are:
- I = Input (Source Deck)
- L = Listable Output (Assembly Listing)
- B = Binary Output (Object Deck)
- O = Library Medium (Basic Assembly, relocatable loader, control program, I/O drivers)

Sample Execution Run



At HLT 0, operator entries to the A register are:
- I = Input (Object Deck)
- L, B = Programmer Usage
- O = Library Medium (Relocatable Loader, etc.)

BASIC UTILTY B

The Autoload Utility System:

- Loads and links routines
- Contains resident routines for tape handling
- Loads and links Central Input/Output (CIO) routines to loaded routines
- Transfers control to routines
- Provides drivers for magnetic tape, card reader, punch, printer, and typewriter
- Provides a limited facility for making unit assignments
- Provides library routines for peripheral processing

Operator intervention is possible through typewriter or console entries as shown below:

TYPEWRITER ENTRIES

If jump key one is not set, the programmer enters control information from the typewriter as follows:

General Form: NAME, parameter list

NAME is the name of a resident routine of the system (e.g. REWIND, UNLOAD) or the name of a routine which has been loaded previously by the resident routine named FETCH (e.g. COPYS, VERIFY). NAME consists of 1 to 8 BCD characters, alphabetic or numeric, excluding commas and periods. A comma follows if a parameter list is applicable. A period follows if there is no parameter list or no more parameters.

Parameter list is a list of the specific parameters required by the routine. See the individual routines which follow:

Resident routines:

REWIND,n.	Rewind Unit n.
UNLOAD,n.	Unload Unit n.
BACKSPCE,n.	Backspace Unit n.
SKFF,n.	Skip one file forward on unit n.

SKFB,n.	Skip one file backward on unit n.
WREOF,n.	Write end of file on n.
ERASE,n.	Erase bad spot on n.
CONTROL,n.	Instructs utility executive to receive next control statement from unit n. If n = TYP, unit is the typewriter; if n = CONSOLE, unit is the console.
DUMP,addr1,addr2,n.	Dumps core from addr1 (octal) to addr2 (octal) on unit n (decimal).
ASSIGN,n,mnemonic.	Assign unit n to hardware designated by the mnemonic.

Mnemonic	Hardware
Mxyz	Magnetic tape, channel x, controller y, unit z.
TPWR	Console typewriter
CDRD	Card reader
CDPU	Card punch
PRNT	Printer

ASSIGN,n,m.	Equate N to unit number m previously designated
CHKDNS,n.	Check the density of unit n.
SETDNS,n,mnemonic.	Set the density of unit n to mnemonic L = low, M = medium, H = high.

FETCH,n,name1,name2,. . . namem. Load and link the named routines and their subroutines from unit n (decimal). Routines must be in relocatable binary format. Multiple calls to FETCH do not destroy previously loaded routines.

FETCH,A.

FETCH,B.

The above two control statements executed in sequence make A and B both available for subsequent execution.

CLEAR.	Restores UTILITY tables to resident routines and stores UJP ABNORMAL throughout available memory.

LIBRARY ROUTINES

COPYS,n1,n2.	Copys 40 word binary records or up to 136 character BCD records from n1 to n2.
COPYT,n1,n2,n3,n4.	Copy n4 records from unit n1 to unit n2; list on logical unit n3. Tape to tape copy.
VERIFY,n1,n2,n3,n4.	Read and match records from unit n1 to unit n2; write offending records on unit n3. If n4 is omitted, one file only is verified. n4 specifies number of records.
COPYWS xxx,n1,n2,n3.	Copies BCD records from n1 (tape or card reader) to n2 and n3 (tape, printer, or punch). Sequence numbers beginning at 00000 and increased by 10 on each succeeding record are placed in columns 76 to 80. xxx is placed in columns 73 to 75.
COPYTSQ,xxxxxxxx,n1,n2,n3.	Copies BCD card images from unit n1 to units n2 and n3 until the sequence identifier, xxxxxxxx, is found in columns 72 through 80. n2 or n3 may be deleted or given the value zero so that one destination tape is used. If both n1 and n2 are deleted or both given the value zero, the input tape is positioned at the record following the record containing the sequence identifier.

CONSOLE ENTRIES

RESIDENT ROUTINES

Control statement:	Console entries:
REWIND	Enter 0 into A, n into B1.
UNLOAD	Enter 1 into A, n into B1.
BACKSPCE	Enter 2 into A, n into B1.
SKFF	Enter 3 into A, n into B1.
SKFB	Enter 4 into A, n into B1.
WREOF	Enter 5 into A, n into B1.
ERASE	Enter 6 into A, n into B1.
DUMP	Enter 7 into A, addr1 into B1, addr2 into B2, and n into B3.
CONTROL	Enter 10_8 into A, n into B1.
FETCH	Enter BCD code for name in AQ (left oriented), n into B1, and 77777 into B2.
ASSIGN	Enter 11_8 into A, n into B1, BCD mnemonic into Q.
CHKDNS	Enter 12_8 into A, n into B1.
SETDNS	Enter 13_8 into A, n into B1, density code into B2. †
CLEAR	Enter 14_8 into A.

LIBRARY FUNCTIONS

Control statement: ††

COPYS	Enter n1 into B1, n2 into B2.
COPYT	Enter n1 into B1, n2 into B2, n3 into B3, n4 into Q.

†† After a FETCH has been executed, the A register contains the index of the loaded routine and B2 contains 77777. Enter index of loaded routine into the A register.

† Density code 1 into B2 indicates high density; code 2 into B2 indicates medium; code 3 into B2 indicates low.

VERIFY	Enter n1 into B1, n2 into B2, n3 into B3, n4 into Q.
COPYWS	Enter BCD mnemonic (right oriented) into AQ, n1 into B1, n2 into B2, n3 into B3.
COPYTSQ	Enter BCD sequence identifier into EQ, n1 into B1, n2 into B2, n3 into B3.

A	=	Register A
AQ	=	Register AQ
B1	=	Index Register 1
B2	=	Index Register 2
B3	=	Index Register 3

Note: If switch one is set, enter information from console, otherwise, enter information from the typewriter.

INDEX

CONTROL DATA SALES OFFICES

ALAMOGORDO • ALBUQUERQUE • ATLANTA • BOSTON • CAPE CANAVERAL
CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS • DAYTON
DENVER • DETROIT • DOWNEY, CALIF. • HONOLULU • HOUSTON • HUNTSVILLE
ITHACA • KANSAS CITY, KAN. • LOS ANGELES • MINNEAPOLIS • NEWARK
NEW ORLEANS • NEW YORK CITY • OAKLAND • OMAHA • PALO ALTO
PHILADELPHIA • PHOENIX • PITTSBURGH • SACRAMENTO • SALT LAKE CITY
SAN BERNARDINO • SAN DIEGO • SEATTLE • WASHINGTON, D.C.

INTERNATIONAL OFFICES

FRANKFURT, GERMANY • HAMBURG, GERMANY • STUTTGART, GERMANY
ZURICH, SWITZERLAND • MELBOURNE, AUSTRALIA • SYDNEY, AUSTRALIA
CANBERRA, AUSTRALIA • ATHENS, GREECE • LONDON, ENGLAND • OSLO, NORWAY
PARIS, FRANCE • STOCKHOLM, SWEDEN • MEXICO CITY, MEXICO, (REGAL
ELECTRONICA DE MEXICO, S.A.) • OTTAWA, CANADA, (COMPUTING DEVICES OF
CANADA, LIMITED) • TOKYO, JAPAN, (C. ITOH ELECTRONIC COMPUTING
SERVICE CO., LTD.)

CONTROL DATA
CORPORATION

8100 34th AVENUE SOUTH, MINNEAPOLIS, MINNESOTA 55440

Pub. No. 60057100

Litho in U.S.A.