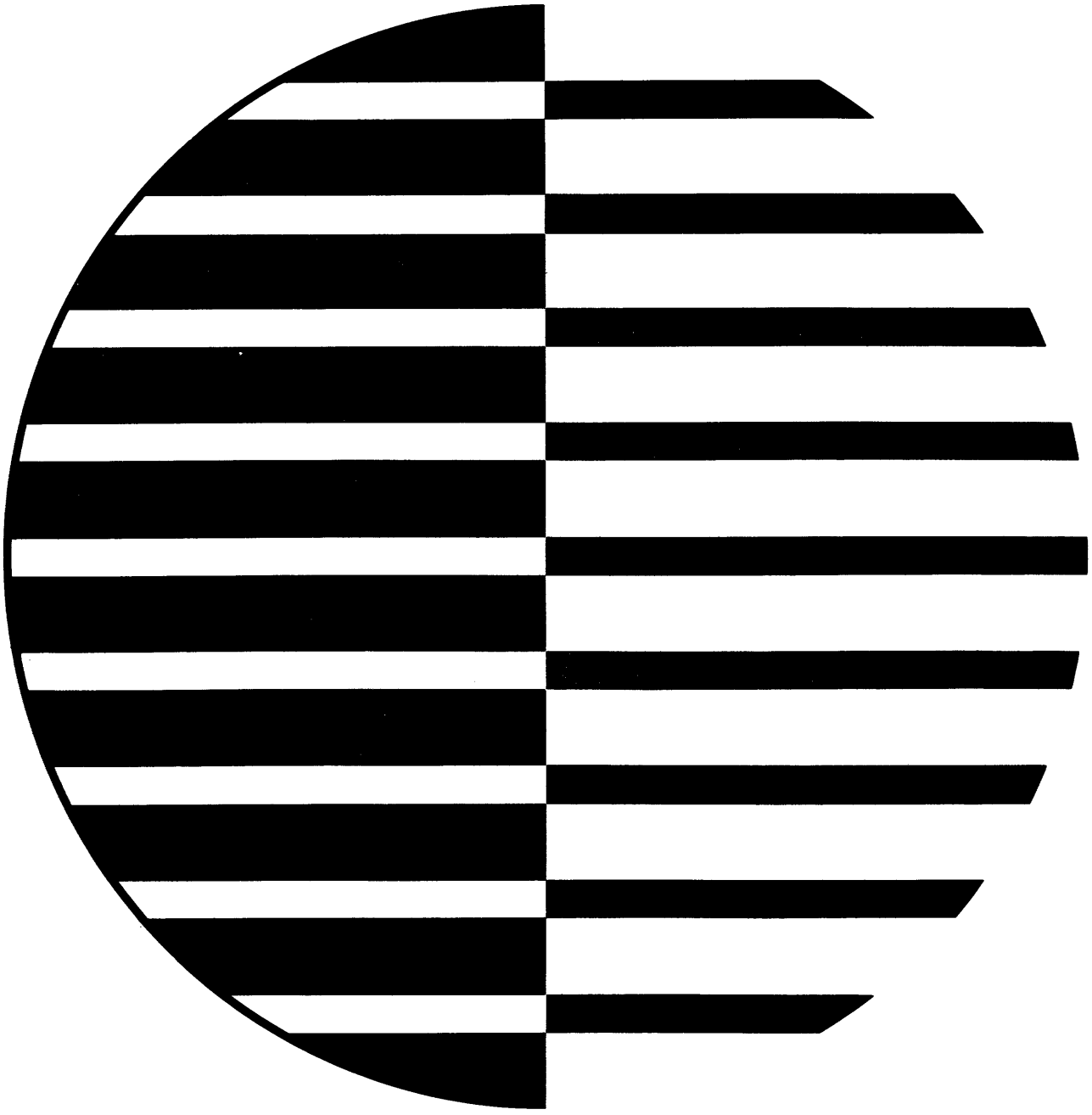
CONTROL DATA® 6000 SERIES COMPUTER SYSTEMS

ASCENT General Information Manual



Additional copies of this manual may be obtained from the nearest Control Data Corporation Sales office listed on the back cover.

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

February, 1966
Pub. No. 60135400

CONTENTS

INTRODUCTION 1

ASCENT, the assembly system for the central processor, is a two-pass assembler executed in the central processor of a Control Data® 6400, 6600, or 6800 computer. Operating under control of the SIPROS system, ASCENT simplifies the writing of machine-language programs for the central processor through mnemonic instructions and symbolic addressing.

In addition, ASCENT offers programming aids, pseudo operation codes, and provisions for the inclusion of system and programmer macros. The principal features of ASCENT are as follows:

- Use of all I/O functions in the operating system
- Use of system and programmer-defined macros
- Ability to mix ASCENT code with the FORTRAN language on a line-for-line basis
- Subroutine calls
- Pseudo commands
- Peripheral processor program calls
- Access to all features of the central processor

SYSTEM CONFIGURATION

Minimum equipment requirement to run ASCENT under control of SIPROS is as follows:

- A 6000-series computer with a central processor, 10 peripheral processors, and 12 data channels
- One disk unit with a storage capacity of 8 million 60-bit words
- One display console
- One 1200 card/minute reader
- One 1000 line/minute printer
- One 250 card/minute punch
- Bank of two 607 or 626 magnetic tape units

LANGUAGE SPECIFICATIONS 2

DEFINITIONS

CHARACTERS

ASCENT uses the following character set:

Letters A through Z

Numbers 0 through 9

+ - / * = ( ) . , $ space

SYMBOLS

A symbol is any arrangement of letters and numbers which starts with a letter and contains no more than eight characters.

CONSTANTS

Constants may be any of the following forms:

Integer	18 or less decimal digits ranging from $-(2^{59}-1)$ to $(2^{59}-1)$.
Octal	20 or less octal digits 0 through 7 with a B suffix
Symbolic	Meets the specifications for a symbol but is equated to a constant or to the difference of two symbols.

SINGLE-PRECISION FLOATING POINT

Expressed by one of two forms:

15 or less decimal digits and a single decimal point.

15 or less decimal digits with or without a single decimal point followed by the power of 10 representation shown below:

E $\pm$ n or En

E specifies that an exponent follows; n, 3 or less decimal digits, is the power of 10 to be applied to the constant.

The sign of n may be omitted if positive.

DOUBLE-PRECISION FLOATING POINT

Expressed by one of two forms:

29 or less decimal digits with or without a decimal point and a letter D suffix.

29 or less decimal digits with or without a decimal point followed by the power of 10 representation as shown below:

D $\pm$ n or Dn

D is a required letter

n, 3 or less decimal digits, is the power of 10 to be applied to the constant

The sign of n may be omitted if positive.

COMPLEX

Any pair of single-precision floating-point constants separated by a comma and enclosed in parentheses.

OPERATORS

In certain cases, operators join register mnemonic codes in defining the numerical operation code of an instruction.

The following operators are used:

+ addition

- subtraction

* multiplication

/ division

The + and - are also used in address manipulation specifications.

LITERALS

Literals are used for addressing a core location whose contents are specified by the value within parentheses. Literals may be any of the following forms:

(Constant)

(Symbol)

(Symbol $\pm$ I)

(Symbol - Symbol)

I is an integer, octal or symbolic constant.

When a two-word form such as

(Floating Double Precision Constant)

(Complex Constant)

is defined, the first word only is addressed.

SEPARATORS

The following separators indicate the end of distinct instruction entities:

$, space . =

Other characters, which may assume the role of separators depending upon usage, are:

+ - * /

OPERANDS

Operands are combinations of symbols, literals, the operators + and -, and certain types of constants. In the acceptable forms shown below, I is an integer, octal or symbolic constant.

Symbol

Symbol ± I

Symbol - Symbol

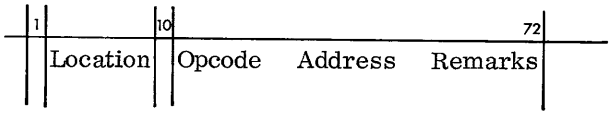
± I

Literal

Literal + I

FORMAT

ASCENT has one basic instruction format with four fields:



ASCENT considers only card columns 2 through 72. Column 1 is reserved for the exclusive use of the Programming System Control Package. A blank in column 10 separates the location and opcode fields.

An input card may contain up to six instructions separated by dollar signs. Only one location field may appear on a card regardless of the number of instructions it contains. It applies only to the first instruction on the card.

The $ acts as the recurrence of column 10; the next expected item is an opcode.

All instructions must be completed prior to column 73.

LOCATION FIELD

This field provides a symbol for referencing by other instructions. The location field is fixed length; it may be blank or contain a symbol starting in column 2-5 and ending prior to column 10.

OPCODE FIELD

The Opcode field defines the instruction: It is variable length, starting in or after column 11 and terminating with at least one separator.

The Opcode field may contain any of the following items:

6000 Series central processor mnemonic codes or their octal equivalents

ASCENT pseudo codes

System macro codes

Programmer defined macro

Integer or octal constant

Mnemonic codes are evaluated to determine their octal equivalents; the octal value is inserted into the instruction word. Pseudo operations are interpreted and used in assembler sequence control. System macros are replaced by calling sequences to a resident communication subroutine.

ADDRESS FIELD

The address field is variable length and has any of the following formats:

REGISTER

—REGISTER

REGISTER OPERATOR REGISTER

—REGISTER OPERATOR REGISTER

REGISTER REGISTER OPERAND

REGISTER OPERATOR OPERAND

REGISTER OPERAND

OPERAND

LIST

List is a sequence of registers and operands as specified for the operation code. Adjacent operands must be separated by a comma. The LIST form in an address field is used in certain pseudo and macro codes.

The content of the address field varies with the instruction. Therefore, several types of instructions are important in its specification:

No address required

Numeric

Registers and Operators

Registers only

Opcode complete, order independent

Opcode complete, order dependent

18-bit arithmetic with registers and operand

18-bit arithmetic for sum of two registers

18-bit arithmetic for difference of two registers

REMARKS FIELD

The remarks field contains programming notes which have no effect on the assembly process. It must begin with a period in or after column 11, or it may be blank.

Example: The instructions shown on the following page are part of a program for the data display unit.

Location	Instruction		Remarks
	BX6	X1	.X1 TO X6
START	BX4	-X3	.-X3 TO X4
	FX7	X6*X4	.FLOATING X6*X4 TO X7
	BX3	-X4+X1	.X1+COMP. X4 TO X3
	EQ	B5 B2 AB	.IF B5=B2, GO TO AB
	SA7	B2+DATA	.STORE X7 TO DATA+B2
	SA7	DATA	.STORE X7 TO BO+DATA
	NZ	X1 ABC	.IF X1 NOT ZERO, GO TO ABC
	RJ	SUB	.RETURN JUMP TO SUB
START 1	RDC	1, ST, (BA), (BA+8), 8, 2	.LIST
	SB1	1 $ SA2 DATA+1	.PACKED CARD
START 2	LX1	6 $ MX2 48 $ JP AB+2 $ SB6 -8 $SA5 B6+DATA $ SB7 B5-B6 (one card)	.MAXIMUM 6 PER CARD .BEGIN REMARKS WITH PERIOD
	JP	B2+BETA	.JUMP TO B2+BETA

PSEUDO OPERATION CODES 3

ASCENT pseudo operation codes, listed below, enable the programmer to control the printing of the assembly listings, perform numerical and alphabetic conversions, reserve storage areas, and specify subroutine names. The instruction format for pseudo operation codes is the same as the basic ASCENT mnemonic coding format:

ASCENT	defines central processor program.
END	defines end of central processor program.
ASPER	defines peripheral processor routine.
SUBROUTINE	defines subroutine name.
BSSD	defines file on a specified disk unit.
BSS	reserves central memory region.
BSSZ	reserves central memory region and sets it to zero.
EQU	equates a symbol to a value.
DPC	inserts display-coded characters into the program.
BCD	inserts BCD characters into the program.
CON	defines constants in the program.
LIST	controls side-by-side output listing.
SPACE	determines line spacing output listing.
EJECT	ejects listing to top of next page.

MACROS 4

PROGRAMMER MACROS

To use a programmer macro, a definition or skelton must precede the first executable instruction in the program. After the macro is defined, it may be used by specifying the macro symbolic name as an operation code and listing the actual parameters. Since the amount of space reserved for the macro symbol table is an installation parameter, there is no restriction on the number of macros that may be used within any one program. System macros may be used within programmer macros.

SYSTEM MACROS

ASCENT provides system macros for the programming of all peripheral devices and for requesting standard system operations. Since SIPROS provides automatic buffering, input and output need not be buffered directly by the programmer. However, through a character (an appended W) associated with each macro, the programmer can specify whether the computations in his program must wait until the macro has been executed, or whether execution can proceed without the macro results. System macros are included for the following operations:

Magnetic tape

Disk

Printer

Card

Console

Segmentation

Request-and-release central processor memory

Request-and-release disk space

Macros are executed as they are encountered in a program. However, printer and punch output is retained on the system disk until the program is completed. Although system macros used in an ASCENT program are actually executed by a peripheral and control processor, an ASCENT programmer need not be aware of this; the macros are automatically routed to the peripheral and control processors by SIPROS.

MAGNETIC TAPE MACROS

RQTW	requests a tape assignment from the system.
DRTW	releases a tape back to the system.
SFFW	searches a tape forward until it detects a file mark.
SFBW	searches the tape backward until it detects a file mark.
WFMW	writes a file mark.
RWLW	rewinds a tape to the load point.
RWUW	rewinds a tape for unloading.
FSPW	spaces a tape forward.
BSPW	backspaces a tape.
RFCW	reads a tape forward in coded mode.
RFBW	reads a tape forward in binary mode.
WRCW	writes a tape in coded mode.
WRBW	writes a tape in binary mode.

DISK MACROS

RDHW	reads a disk record and holds the data on disk.
RDRW	reads a disk record and releases the data on disk.
WRDW	writes a record on disk.

PRINTER MACROS

SSPW	single spaces printer.
DSPW	double spaces printer.
FC7W	selects format channel 7.
FC8W	selects format channel 8.
MC1W	selects monitor channel 1.
MC2W	selects monitor channel 2.
MC3W	selects monitor channel 3.
MC4W	selects monitor channel 4.
MC5W	selects monitor channel 5.
MC6W	selects monitor channel 6.
CMCW	clears monitor channel 1-6.
SPAW	suppresses space after next print.
PRNW	prints single line or multiple lines.

CARD MACROS

PCHW	punches cards.
RDCW	reads cards.

DISPLAY MACROS

DRSW	displays on right scope for system time limit.
DSLW	displays on left scope for system time limit.
DHRW	displays on right scope and holds indefinitely.
DHLW	displays on left scope and holds indefinitely.
RDPW	removes display.
RTYW	reads console typewriter.

SYSTEM ACTION MACROS

TPPW	transfers a PP program from central memory to a peripheral processor and starts executing the program with the first ASPER instruction.
RQMW	requests central memory.
DRMW	releases central memory.
RQDW	requests disk space.
DRDW	releases disk space.

WAIT MACRO

WAIW	checks status of the specified input-output routine and exits on a request completed or aborted condition.

OVERLAY MACRO

OLAY	defines and loads a normal segment at the central memory location associated with the specified overlay point. An error return entry may be specified for abnormal loader termination.

SYSTEM LIBRARY SUBROUTINES

A set of subroutines is included in the system library for general use by ASCENT and FORTRAN-66. In many cases, the library routines are referenced as function subroutines by FORTRAN-66 and as subroutines in ASCENT coding. Therefore, a compatible format is used in the definition of the routines, the FORTRAN code generators, and the ASCENT calling sequences.

The general form is as follows:

CALL name (list)

CALL	is a FORTRAN statement.
name	is the name of a routine.
list	contains a sequence of operands which define actual parameters.

PROGRAMMER DEFINED SUBROUTINES

In addition to the library functions, a programmer may define new subroutines in the process of writing a program. Symbols within a subroutine are local to that subroutine.

A compatible definition and calling format are used. To define a subroutine, a header card is needed:

subroutine symbol (list)

subroutine	is the pseudo operation code.
symbol	is the identification name for the subroutine.
list	is a sequence of symbols, called formal parameters, separated by commas, which represent I/O variables to the subroutine.

References to the subroutine are made with the statement:

CALL symbol (list)

symbol — is the same combination of letters used in the subroutine identification name.

list — contains the names and values of the I/O parameters for the subroutine in the same order as given in the subroutine definition list.

Other communications between the subroutine and the calling programs include: alternate entry points, common data blocks, and variable subroutine and function names for calls made within the subroutine.

Subroutines may be assembled independently of other subroutines and the calling programs. Subroutine linkage and common variable references are adjusted at load time.

ASCENT ERROR PRINTOUTS

A Literal Table Full. The literal is not assigned a location.

B Symbol Table Full. The symbol is not assigned a location.

D Duplicate Symbol. The symbol in the location field has been previously defined. A list of all duplicate symbols is printed at the end of the side-by-side listing.

E Instruction Error. There are more than six instructions on the card.

F Format Error. An error is detected in the format of an instruction.

I Integer Error. An error is detected in a decimal or octal number.

K K-Field Error (address field). The address portion of the instruction does not meet program specifications or is out of range.

L Literal Error. An error is detected in the evaluation or conversion of the literal.

M Multiply Defined Reference. A reference is made to a symbol that appears more than once in the location field.

O Operation Code Error. The operation code cannot be evaluated. ASCENT assumes an operation code of zero and processes the instruction accordingly.

P Parameter List Error. The parameter list does not satisfy ASCENT specifications - too few or too many parameters.

R Register Error. An error is detected in the format of a register name or its improper usage.

S Sign Error. A sign is incorrect or out of order.

T Tag Error. A symbol in the location or address field does not meet ASCENT specifications.

U Undefined Symbol. A reference is made to a symbol that does not appear in the location field. ASCENT assigns a location at the end of the object program to each unique undefined symbol. In certain pseudo codes (EQU, BSS, BSSZ), a symbol used in the address field must be defined prior to the pseudo code. A list of undefined symbols appears at the end of the side-by-side listing.

PROGRAM SEGMENTATION 7

Any ASCENT program may be divided into segments to reduce the amount of memory space required. When a program has been segmented, one part, called the basic segment, must remain in memory at all times. The remaining segments, called normal segments, may be loaded into memory by a system macro, OLAY, at any time.

CP PROGRAM SEGMENTATION

The segmentation scheme for central processor programs allows the programmer to define segments either at load time or execute time and to specify the routines to be included in the basic segment. It also allows multiple segments to be loaded into memory. With regard to the definition and use of central memory, three options are available:

The system releases all unused memory after the basic segment is loaded; it then requests and releases memory as required in handling overlays.

The system releases all unused memory after the first occurrence of the OLAY macro; it then requests and releases memory as required in handling overlays.

The system does not release memory at any time; if there is not sufficient memory for an overlay, additional memory is requested to fulfill the overlay request.

The option selected is defined in the basic segment definition card.

A FORTRAN library routine, SEGMAP, facilitates checkout of a segmented program. This routine inserts a map of the current contents of memory into a specified print file. The map provides such information as the contents of each overlay, the names of routines called but not loaded, and so forth.

SEGMENT DEFINITION

The basic segment is defined by a basic segment definition card supplied by the programmer; it may be made up of a main program, subroutines, and other segments. The name of the routine in the basic segment that is to initially receive control must be enclosed in asterisks. If this routine is also defined as a normal segment, the normal segment name must also be enclosed in asterisks.

The same conventions used in defining the basic segment are used in defining normal segments. In addition, however, a normal segment may be defined in the program by the OLAY macro.

SEGMENT LOADING

Only the routines defined in the basic segment card and the standard library routines called by the basic segment routines are loaded into memory when the basic segment is loaded. Calls to non-standard library routines from the basic segment are filled with error stops and given the proper addresses after the called routines are loaded into memory.

The only routines loaded into memory by an OLAY macro are those named in the macro list and the standard library routines called by them. A routine already in memory will not be reloaded. Calls to non-standard library routines are filled with error stops and given the proper addresses after the called routines are loaded.

During the loading of an overlay, control is normally returned to the instruction immediately following the OLAY macro. However, it is possible to switch control directly into the new overlay by enclosing in asterisks the name of the routine that will receive control. If the routine that is to receive control appears in the OLAY macro list, the name must be enclosed in asterisks in the list. If it also appears in a segment definition card, it must also be enclosed in asterisks on the card.

CENTRAL PROCESSOR OPERATION CODES

Octal Opcode	Mnemonic	Address	Meaning
			BRANCH UNIT
00	PS		Program stop
01	RJ	K	Return jump to K
02	JP	Bi + K	Jump to Bi + K
030	ZR	Xj K	Jump to K if Xj = 0
031	NZ	Xj K	Jump to K if Xj $\neq$ 0
032	PL	Xj K	Jump to K if Xj = positive
033	NG	Xj K	Jump to K if Xj = negative
034	IR	Xj K	Jump to K if Xj is in range
035	OR	Xj K	Jump to K if Xj is out of range
036	DF	Xj K	Jump to K if Xj is definite
037	ID	Xj K	Jump to K if Xj is indefinite
04	EQ	Bi Bj K	Jump to K if Bi = Bj
04	ZR	Bi K	Jump to K if Bi = B0
05	NE	Bi Bj K	Jump to K if Bi $\neq$ Bj
05	NZ	Bi K	Jump to K if Bi $\neq$ B0
06	GE	Bi Bj K	Jump to K if Bi $\geq$ Bj
06	PL	Bi K	Jump to K if Bi $\geq$ B0
07	LT	Bi Bj K	Jump to K if Bi < Bj
07	NG	Bi K	Jump to K if Bi < B0

Octal Opcode	Mnemonic	Address	Meaning
			BOOLEAN UNIT
10	BXi	Xj	Transmit Xj to Xi
11	BXi	Xj*Xk	Logical product of Xj & Xk to Xi
12	BXi	Xj + Xk	Logical sum of Xj & Xk to Xi
13	BXi	Xj - Xk	Logical difference of Xj & Xk to Xi
14	BXi	- Xk	Transmit the comp. of Xk to Xi
15	BXi	- Xk*Xj	Logical product of Xj & Xk comp. to Xi
16	BXi	- Xk + Xj	Logical sum of Xj & Xk comp. to Xi
17	BXi	- Xk - Xj	Logical difference of Xj & Xk comp. to Xi
			SHIFT UNIT
20	LXi	jk	Left shift Xi, jk places
21	AXi	jk	Arithmetic right shift Xi, jk places
22	LXi	Bj Xk	Left shift Xk nominally Bj places to Xi
23	AXi	Bj Xk	Arithmetic right shift Xk nominally Bj places to Xi
24	NXi	Bj Xk	Normalize Xk in Xi and Bj
25	ZXi	Bj Xk	Round and normalize Xk in Xi and Bj
26	UXi	Bj Xk	Unpack Xk to Xi and Bj
27	PXi	Bj Xk	Pack Xi from Xk and Bj
43	MXi	jk	Form mask in Xi, jk bits
			ADD UNIT
30	FXi	Xj + Xk	Floating sum of Xj and Xk to Xi
31	FXi	Xj - Xk	Floating difference Xj and Xk to Xi
32	DXi	Xj + Xk	Floating DP sum of Xj and Xk to Xi
33	DXi	Xj - Xk	Floating DP difference of Xj and Xk to Xi

Octal Opcode	Mnemonic	Address	Meaning
34	RXi	Xj + Xk	Round floating sum of Xj and Xk to Xi
35	RXi	Xj - Xk	Round floating difference of Xj and Xk to Xi
			LONG ADD UNIT
36	IXi	Xj + Xk	Integer sum of Xj and Xk to Xi
37	IXi	Xj - Xk	Integer difference of Xj and Xk to Xi
			MULTIPLY UNIT
40	FXi	Xj * Xk	Floating product of Xj and Xk to Xi
41	RXi	Xj * Xk	Round floating product of Xj & Xk to Xi
42	DXi	Xj * Xk	Floating DP product of Xj & Xk to Xi
			DIVIDE UNIT
44	FXi	Xj / Xk	Floating divide Xj by Xk to Xi
45	RXi	Xj / Xk	Round floating divide Xj by Xk to Xi
46	NO		No operation
47	CXi	Xk	Count the number of 1's in Xk to Xi
			INCREMENT UNIT
50	SAi	Aj + K	Set Ai to Aj + K
50	SAi	Aj - K	Set Ai to Aj + comp. of K
51	SAi	Bj + K	Set Ai to Bj + K
51	SAi	Bj - K	Set Ai to Bj + comp. of K
52	SAi	Xj + K	Set Ai to Xj + K
52	SAi	Xj - K	Set Ai to Xj + comp. of K

Octal Opcode	Mnemonic	Address	Meaning
53	SAi	Xj + Bk	Set Ai to Xj + Bk
54	SAi	Aj + Bk	Set Ai to Aj + Bk
55	SAi	Aj - Bk	Set Ai to Aj - Bk
56	SAi	Bj + Bk	Set Ai to Bj + Bk
57	SAi	Bj - Bk	Set Ai to Bj - Bk
60	SBi	Aj + K	Set Bi to Aj + K
60	SBi	Aj - K	Set Bi to Aj + comp. of K
61	SBi	Bj + K	Set Bi to Bj + K
61	SBi	Bj - K	Set Bi to Bj + comp. of K
62	SBi	Xj + K	Set Bi to Xj + K
62	SBi	Xj - K	Set Bi to Xj + comp. of K
63	SBi	Xj + Bk	Set Bi to Xj + Bk
64	SBi	Aj + Bk	Set Bi to Aj + Bk
65	SBi	Aj - Bk	Set Bi to Aj - Bk
66	SBi	Bj + Bk	Set Bi to Bj + Bk
67	SBi	Bj - Bk	Set Bi to Bj - Bk
70	SXi	Aj + K	Set Xi to Aj + K
70	SXi	Aj - K	Set Xi to Aj + comp. of K
71	SXi	Bj + K	Set Xi to Bj + K
71	SXi	Bj - K	Set Xi to Bj + comp. of K
72	SXi	Xj + K	Set Xi to Xj + K
72	SXi	Xj - K	Set Xi to Xj + comp. of K
73	SXi	Xj + Bk	Set Xi to Xj + Bk
74	SXi	Aj + Bk	Set Xi to Aj + Bk
75	SXi	Aj - Bk	Set Xi to Aj - Bk
76	SXi	Bj + Bk	Set Xi to Bj + Bk
77	SXi	Bj - Bk	Set Xi to Bj - Bk

6000 SERIES PUBLICATIONS

TITLE	PUB. NO.
General	
6600 Parallel Operation	55-0122
6600 Remote Time-Sharing	55-0123
6600 Code Book	60141900
Hardware	
6000 Series, Reference Manual	60045000
6000 Series, Site Preparation and Installation Manual	60143400
6000 Series, Input/Output Specification	60064100
6060 Remote Calculator, Reference Manual	60147900
Software	
6000 Series, ASCENT, General Information Manual	60135400
6000 Series, ASPER, General Information Manual	60135100
6000 Series, FORTRAN 66, General Information Manual	60135500
6000 Series, SIMSCRIPT, General Information Manual	60133200
6000 Series, SIPROS, General Information Manual	60135600
6000 Series, File Manager, General Information Manual	60135000
6000 Series, PERT, General Information Manual	60133300
6000 Series, Report Generator, General Information Manual	60135800
6000 Series, PERT/TIME Reference Manual	60133600
6000 Series, Chippewa FORTRAN, Reference Manual	60132700
6000 Series, Chippewa Operating System Reference Manual	60134400
6000 Series, Matrix Algebra Subroutine Reference Manual	60135200
6000 Series, SIPROS, Operator's Guide	60135700
6000 Series, Statistical Subroutine Reference Manual	60135300
6000 Series, SIPROS-IMPORT 31/EXPORT Reference Manual	38791600
6600 Programming Systems	55-0124
6600 Library Functions	360114500
6600 Standard Programming Packages	PRI Ref or 88-0089
6600 Extended Programming Package	No publication no.
6600 ASCENT, Reference Manual	60101600
6600 ASPER, Reference Manual	60101700
FORTRAN 66, Reference Manual	60101500
6600 SIPROS, Reference Manual	60101800
6000 Series Instant Chippewa Operating System FORTRAN	60133400
6000 Series Instant Chippewa Operating System ASCENT and ASPER	60134500

The above publications may be ordered from the nearest Control Data Corporation Sales office.

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

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

FOLD FOLD

STAPLE STAPLE

CONTROL DATA
CORPORATION

COMMENT AND EVALUATION SHEET

6000 SERIES COMPUTER SYSTEMS
ASCENT General Information Manual

Pub. No. 60135400 February, 1966

YOUR EVALUATION OF THIS MANUAL WILL BE WELCOMED BY CONTROL DATA CORPORATION. ANY ERRORS, SUGGESTED ADDITIONS OR DELETIONS, OR GENERAL COMMENTS MAY BE MADE BELOW. PLEASE INCLUDE PAGE NUMBER REFERENCE.

FROM NAME: _____

BUSINESS ADDRESS: _____

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE