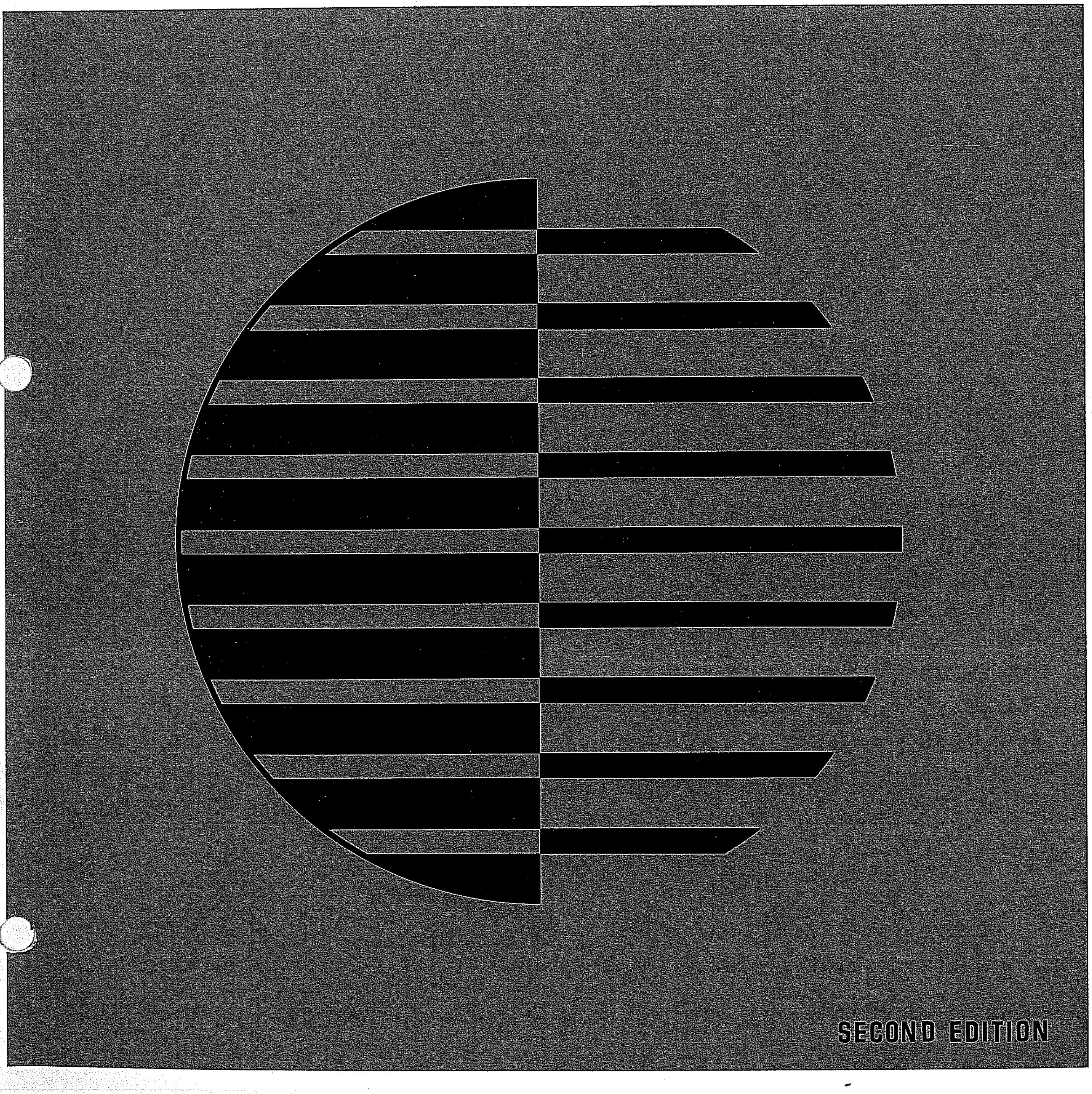
CONTROL DATA® 6600 Computer System Programming System/ Reference Manual

Volume 2 ASPER

Assembly System PERipheral Processor



SECOND EDITION

Peripheral and Control Processor Instruction Execution Times

Mnemonic & Octal Code		Name	Time (Major Cycles)
PSN	00	Pass	1
LJM	01	Long jump to m + (d)	2-3
RJM	02	Return jump to m + (d)	3-4
UJN	03	Unconditional jump d	1
ZJN	04	Zero jump d	1
NJN	05	Nonzero jump d	1
PJN	06	Plus jump d	1
MJN	07	Minus jump d	1
SHN	10	Shift d	1
LMN	11	Logical difference d	1
LPN	12	Logical product d	1
SCN	13	Selective clear d	1
LDN	14	Load d	1
LCN	15	Load complement d	1
ADN	16	Add d	1
SBN	17	Subtract d	1
LDC	20	Load dm	2
ADC	21	Add dm	2
LPC	22	Logical product dm	2
LMC	23	Logical difference dm	2
PSN	24	Pass	1
PSN	25	Pass	1
EXN	26	Exchange jump	min. 20
RPN	27	Read program address	1
LDD	30	Load (d)	2
ADD	31	Add (d)	2
SBD	32	Subtract (d)	2
LMD	33	Logical difference (d)	2
STD	34	Store (d)	2
RAD	35	Replace add (d)	3
AOD	36	Replace add one (d)	3
SOD	37	Replace subtract one (d)	3
LDI	40	Load ((d))	3
ADI	41	Add ((d))	3
SBI	42	Subtract ((d))	3
LMI	43	Logical difference ((d))	3
STI	44	Store ((d))	3
RAI	45	Replace add ((d))	4
AOI	46	Replace add one ((d))	4
SOI	47	Replace subtract one ((d))	4
LDM	50	Load (m + (d))	3-4
ADM	51	Add (m + (d))	3-4
SBM	52	Subtract (m + (d))	3-4
LMM	53	Logical difference (m + (d))	3-4
STM	54	Store (m + (d))	3-4
RAM	55	Replace add (m + (d))	4-5
AOM	56	Replace add one (m + (d))	4-5
SOM	57	Replace subtract one (m + (d))	4-5
CRD	60	Central read from (A) to d	min. 6
CRM	61	Central read (d) words from (A) to m	5 plus 5/word
CWD	62	Central write to (A) from d	min. 6
CWM	63	Central write (d) words to (A) from m	5 plus 5/word
AJM	64	Jump to m if channel d active	2
IJM	65	Jump to m if channel d inactive	2
FJM	66	Jump to m if channel d full	2
EJM	67	Jump to m if channel d empty	2
IAN	70	Input to A from channel d	2
IAM	71	Input (A) words to m from channel d	4 plus 1/word
OAN	72	Output from A on channel d	2
OAM	73	Output (A) words from m on channel d	4 plus 1/word
ACN	74	Activate channel d	2
DCN	75	Disconnect channel d	2
FAN	76	Function (A) on channel d	2
FNC	77	Function m on channel d	2

PREFACE

The 6600 Assembly System for the Peripheral Processors, ASPER, is one of the functional subsets of the 6600 Programming System. It is designed for the 6600 Programming System. It is designed for utility both as a processor of programs written for isolated use in a peripheral processor and as a processor of programs written to work jointly and in synchronization with central processor programs.

ASPER provides a full set of machine mnemonics, pseudo codes and other assembly features. In addition, as a part of the 6600 Programming System, it provides an extended capability in system-oriented features. Some of these features are:

1. Access to all symbols of the central processor program with which it is associated. This symbolic access holds whether the central program is written in ASCENT or FORTRAN or both and it includes access to variables in COMMON storage and subroutine formal parameter list.
2. Reservation of its own central memory words or blocks.
3. Peripheral processor program overlay capability.
4. A full set of system macro instructions for requesting other peripheral processors to handle standard input/output operations.
5. System macros to request loading of other peripheral processor programs.

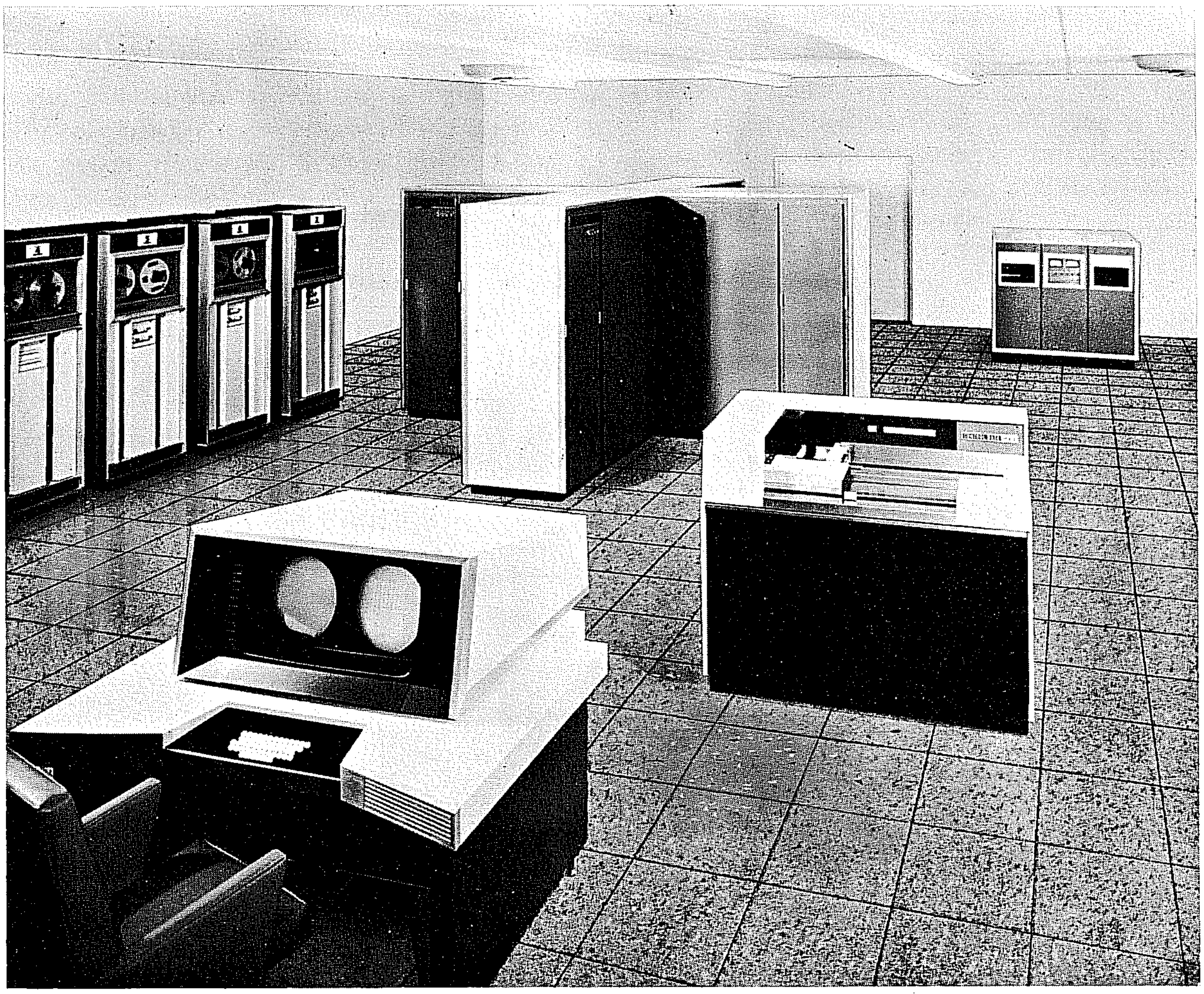
Many of the properties of mixed language programs and program organization are described in detail in Volume I of the 6600 Programming System Manual "ASCENT." Volume II describes the language forms and organization within a defined ASPER routine.

Section 1 of this manual defines specific entities of the symbolic language. Sections 2 and 3 give the instruction forms; Section 4, the pseudo operations; Sections 5 and 6, the system macros; and Section 7, the assembler diagnostics. Section 8 describes peripheral processor program segmentation.

TABLE OF CONTENTS

FIGURES

APPENDIX



6600 COMPUTING SYSTEM

Main frame *(center) — contains 10 peripheral and control processors, central processor, central memory, some I/O synchronizers.*

Display console *(foreground) — includes a keyboard for manual input and operator control, and two 10-inch display tubes for display of problem status and operator directives.*

CONTROL DATA 607 tapes *(left front) — ½ inch magnetic tape units for supplementary storage; binary or BCD data handled at 200, 556, or 800 bpi.*

CONTROL DATA 626 tapes *(left rear) — 1-inch magnetic tape units for supplementary storage; binary data handled at 800 bpi.*

Disk file *(right rear) — Supplementary mass storage device holds 500 million bits of information.*

CONTROL DATA 405 card reader *(right front) — reads binary or BCD cards at 1200 card per minute rate.*

SYSTEM ORGANIZATION

The CONTROL DATA® 6600 is a large-scale, solid-state, general-purpose digital computing system. The advanced design techniques incorporated in the system provide for extremely fast solutions to data processing, scientific and control center problems.

Within the 6600 are eleven independent computers (Fig. 1). Ten of these are constructed with the peripheral and operating system in mind. These ten have separate memory and can execute programs independently of each other or the central processor. The eleventh computer, the central processor, is a very high-speed arithmetic device. The common element between these computers is the large central memory.

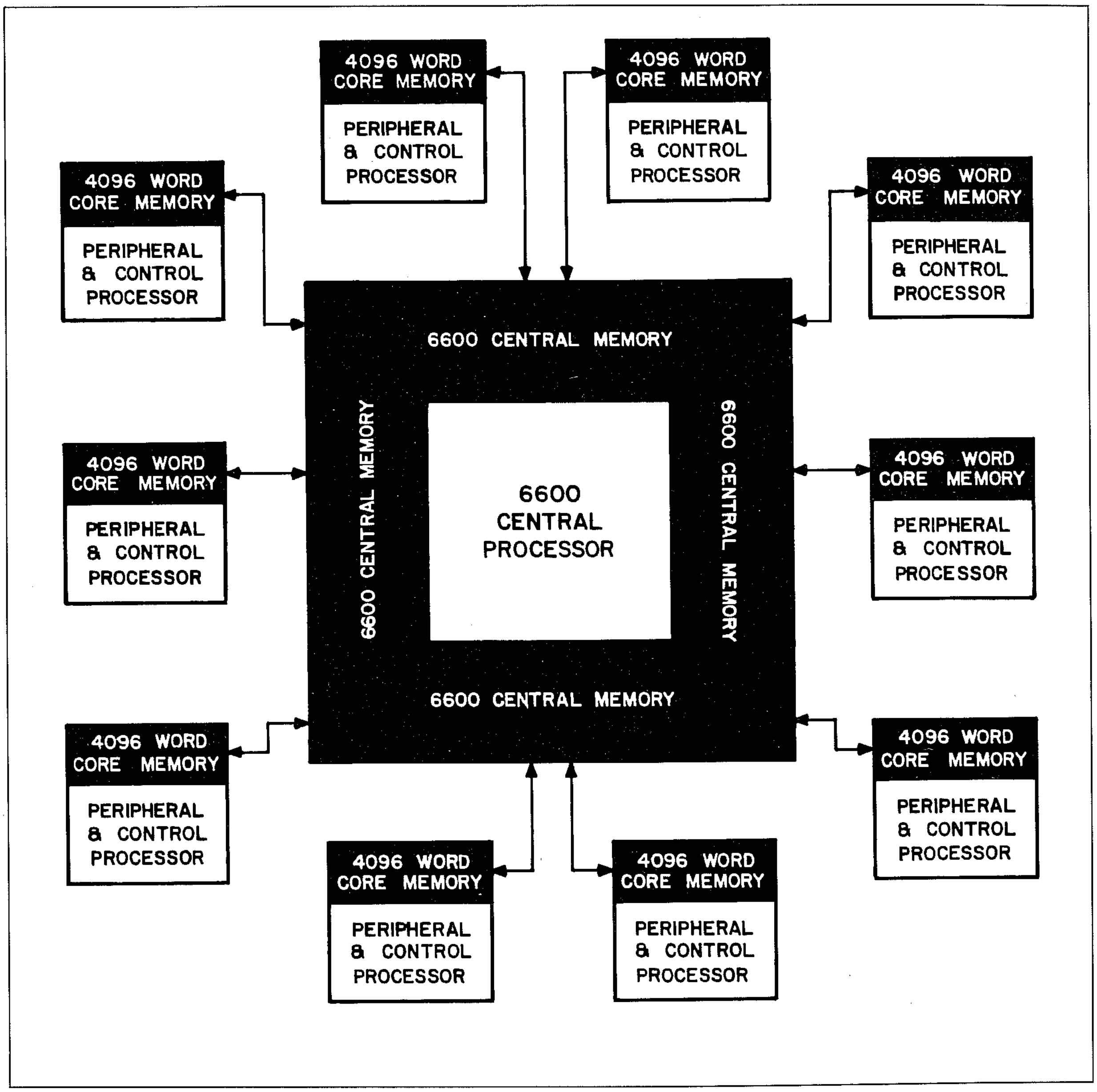
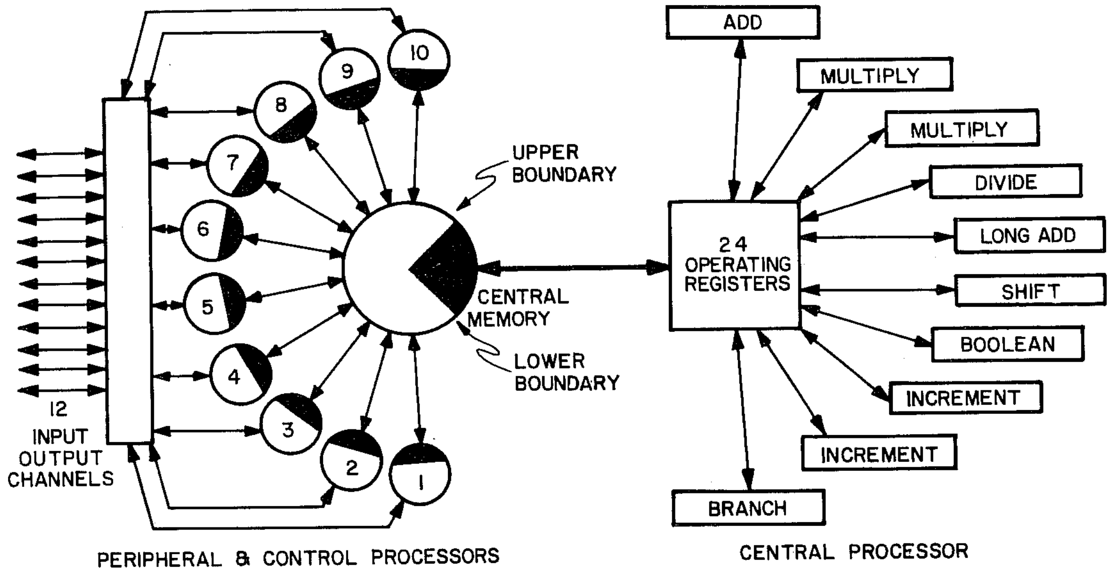


Figure 1 CONTROL DATA 6600



CENTRAL MEMORY

– 131,072 words

– 60-bit words

– Memory organized in 32 logically independent banks of 4096 words with corresponding multi-phasing of banks

– Random access, coincident-current, magnetic core

– One major cycle for read-write

– Maximum memory reference rate to all banks – one address/minor cycle

– Maximum rate of data flow to/from memory – one word/minor cycle

DISPLAY CONSOLE

– Two display tubes

– Modes
Character
Dot

– Character size
Large – 16 characters/line
Medium – 32 characters/line
Small – 64 characters/line

– Characters
26 alphabetic
10 numeric
11 special

Figure 2 BLOCK DIAGRAM OF 6600

1. LANGUAGE DEFINITIONS

1.1 CHARACTERS

ASPER uses the following character set:

The alphabet	Letters A through Z
The arabic numerals	numbers 0 through 9
The special characters	+ − / * = ( ) . , $ space

1.2 SYMBOLS

A symbol is any arrangement of letters and numbers which starts with a letter and contains no more than 8 total characters. The special character * has momentary properties of a symbol under certain usage as defined under 2.2.3.

Examples: T, PROG, ZIZ, ABCD1234

1.3 CONSTANTS

Constants may be any of the following forms:

1.3.1 INTEGER

An integer constant is any arrangement of 4 or less decimal digits $\leqq 2^{12} - 1$.

Examples: 3, 4092, 82

1.3.2 OCTAL

An octal constant is any arrangement of 4 or less octal digits 0 through 7 appended with the letter B.

Examples: 47B, 7770B, 140B

1.3.3 SYMBOLIC

A symbolic constant meets the specifications for a symbol but is equated to a constant or to the difference of two symbols.

Examples:

```
TAM   EQU   3677B−150B
GAT   EQU   64+99
MAG   EQU   20
SAG   EQU   TAG−SAM
```

If TAG and SAM are asigned memory locations 120_8 and 100_8, respectively, then SAG is equated to 20_8.

1.3.4 CP CONSTANT

A central program constant is an 18-bit number $\leqq 2^{17} - 1$.

Examples: 12345B, 131000, 77101B

1.4 OPERATORS

Operators are used in address manipulations only. The two used are:

+ addition

− subtraction

1.5 LITERALS

Literals may be used only in system macros for addressing a word whose contents are specified by the quantity within the parentheses. Only central memory symbols may be used in literals and may be either of the following forms:

(Symbol)

(Symbol ± Constant)

Examples: (CMTAG), (CMTAG-5), (CMTAG+10B)

1.6 SEPARATORS

Separators are used to indicate the end of distinct entities of an instruction. The five characters used are:

space , $. =

1.7 OPERANDS

Operands are combinations of symbols, operators, and constants and are terminated by a separator. Any group of symbols, operators, and constants intended to be a single operand must not contain embedded separators. The acceptable forms are:

Symbol

Symbol±Constant

Symbol−Symbol

± Constant

Restrictions:

1. The form SYMBOL−SYMBOL is restricted to both symbols being ASPER symbols or both being CP Symbols. A mix is undefined.
2. The form SYMBOL±CONSTANT is restricted to constants $\leqq 2^{12} - 1$ for ASPER symbols. For CP symbols, the constant must be $\leqq 2^{17} - 1$.

1.8 FIELDS

An instruction is a combination of the following fields:

LOCATION: Provides a symbol for referencing by other instructions.

OPCODE: Defines the instruction.

ADDRESS: Supplies the instruction with appropriate operands.

REMARKS: Programmer notes only. This field has no effect on the assembly process and must begin with a period in or after column 11.

2. LANGUAGE SPECIFICATIONS

2.1 FORMATS

ASPER has one basic instruction format:

LOCATION OPCODE ADDRESS REMARKS

The Location field is a fixed length field and occupies columns 2-9 on the ASPER input card.

The Opcode field is variable length and starts in or after column 11 on the input card and must be terminated by at least one separator.

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

OPERAND
OPERAND OPERAND
LIST

(LIST is a sequence of operands as specified for the operation code and is used in certain pseudo opcodes and macros. Adjacent operands are separated by commas, spaces, or equal signs.)
The Remarks field is either blank or starts with the special character, period, in any column 11-72.

ASPER considers only card columns 2 through 72. Column 1 is reserved for the exclusive use of the Programming System Control Package. Column 10 is blank and serves as a separator between the location field and the opcode field.

Up to six instructions may be placed on an input card. The special character $ is used to denote the beginning of a new instruction. The following rules apply:

1. Only one location field can be used on a card regardless of the number of instructions it contains. When it is used, it applies to the first instruction on the card.
2. The $ acts as the recurrence of column 10 on the card. The next expected item is an opcode.
3. All instructions on the card must be completed prior to column 73.

2.2 FIELDS

2.2.1 LOCATION FIELD

The location field may be blank or contain a symbol starting in any column 2-5 and ending prior to column 10.

Rules:

1. There may be no duplicate symbols in the location fields within a routine, or between the ASPER routine and the central processor routine of which it is a part. No conflict exists by using symbols in different ASPER routines defined as parts of the same central processor routine.
2. The special character * may not appear in the location field.

2.2.2 OPCODE FIELD

The Opcode field may contain any of the following items:

1. The 6600 peripheral processor mnemonic codes or their octal equivalents as given in Table 1.
2. ASPER pseudo codes as given in Table 2.
3. SYSTEM macro codes as given in Table 3.

Mnemonic codes are evaluated to determine their octal equivalents and 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.

A separator terminates the field.

2.2.3 ADDRESS FIELD

The address field content and length varies with the type of instruction. The different types are described separately.

1. Six-bit address:

   No address – mnemonic ends in N

   Direct – nmemonic ends in D

   Indirect – mnemonic ends in I

   Instructions of this class require only one peripheral memory location. The address field is restricted to one operand. The operand evaluation must produce an octal equivalent $\leqq 77_8$. This value is inserted with the opcode into the instruction word.

2. Twelve-bit address and six-bit index designator:

   Memory – mnemonic ends in M

The operand allowed in the address field is as follows:

OPCODE   ADDRESS   INDEX

Instructions of this class require two peripheral memory locations, the first of which contains the 6-bit opcode and the index designator ($I \leq 77_8$). The second word contains the address portion evaluated to $\leq 2^{12} - 1$. The index designator may be blank.

3. Eighteen-bit address:

Constant – mnemonic ends in C

Instructions of this class require two peripheral memory locations. The first word contains the 6-bit opcode and the high-order 6 bits of the operand. The low-order 12 bits are placed in the second word. The address field is restricted to one operand, the evaluation of which must produce an octal equivalent $\leq 2^{18} - 1$.

SPECIAL USAGE – The special character *, when used as an operand or part of an operand, assumes the value of the current object code address. The legal forms are:

*

* ± Constant

Examples:

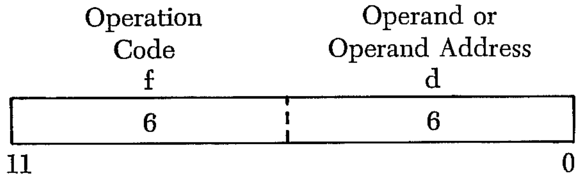
```
LDC   *        .A = current location
UJN   * -4     .Jump back 4 words
```

3. PERIPHERAL PROCESSOR INSTRUCTIONS

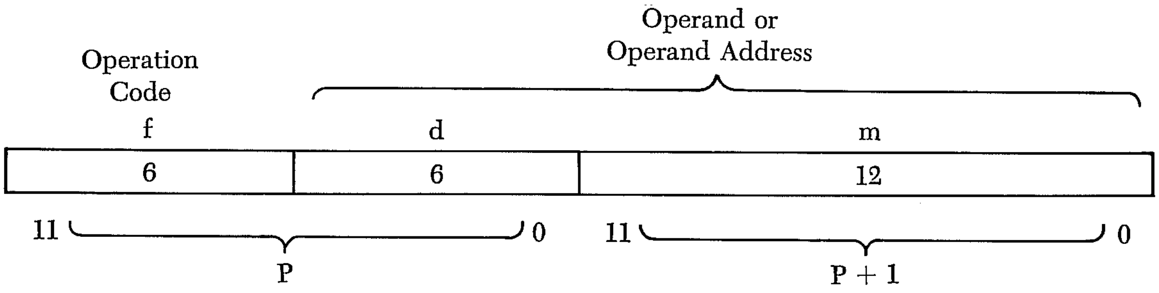
3.1 INSTRUCTION FORMAT

A PP instruction may have a 12-bit or a 24-bit format. The two instruction formats provide for 6-bit or 18-bit operands and for 6-bit, 12-bit or 18-bit operand addresses.

The 12-bit format has a 6-bit operation code, f, and a 6-bit operand or operand address, d.



The 24-bit format requires two memory words. The 6-bit quantity, d, of the first word is used with the 12-bit quantity, m, of the next consecutive word to form an 18-bit operand or operand address.



3.2 ADDRESS MODES

The usage of the quantities d and/or m varies with the addressing mode of an instruction. The three available addressing modes are no address, direct address, and indirect address.

In the no address mode d or dm is taken directly as an operand, thus eliminating the need for storing many constants. By definition, d is a 6-bit quantity $00\text{-}77_8$, but it may be considered as a 12-bit number whose upper 6 bits are zero. The dm quantity consists of 18-bits with d as the upper 6 bits and m as the lower 12 bits.

In the direct address mode, d or $m+(d)$ is used as the operand address. The d quantity specifies one of the first 64 memory locations ($0000\text{-}0077_8$). The $m+(d)$ quantity generates a 12-bit address for referencing all possible PP memory locations ($0000\text{-}7777_8$). If $d = 0$, m is taken as the operand address. If $d \neq 0$, the content of location d is added to m to produce an operand address (indexed direct addressing).

In the indirect address mode, d specifies a location whose content is the address of the desired operand.

Indirect addressing and indexed direct addressing require an additional memory reference over direct addressing.

Examples of the three modes of addressing follow:

Given: d = 25

m = 100

contents of loc. 25 = 0150

contents of loc. 150 = 7776

contents of loc. 250 = 1234

Then:

MODE	INSTRUCTION	A REGISTER
No Address	LDN d	000025
	LDC dm	250100
Direct Address	LDD (d)	000150
	LDM (m + (d))	001234
Indirect Address	LDI ((d))	007776

3.3 DESCRIPTION OF OPERATION CODES

00 PSN Pass

A no operation instruction.

01 LJM m + (d) Long Jump

Jumps to the sequence beginning at the address given by m + (d). If d = 0, then m is not modifiied.

02 RJM m + (d) Return Jump

Jumps to the sequence beginning at the location given by m + (d). If d = 0, then m is not modified. The current program address plus two (P + 2) is stored at the jump address and is used as the return address to the main routine when the sequence is finished.

03 UJN d Unconditional Jump

Provides an unconditional jump of up to 31 steps forward or backward from the current program address, depending on the value of d. If d is positive (01-37$_8$), the jump is forward. If d is negative (40-76$_8$) the jump is backward. The program stops when d equals 00 or 77.

04 ZJN d Zero Jump

Provides a conditional jump of up to 31 steps forward or backward from the current program address if the content of the A register is zero. If A is nonzero, the next instruction is executed. Negative zero (777777) is treated as nonzero. See instruction 03 for an interpretation of d.

05 NJN d Nonzero Jump

Provides a conditional jump of up to 31 steps forward or backward from the current program address if the content of the A register is nonzero. If A is zero, the next instruction is executed. Negative zero (777777) is treated as nonzero. See instruction 03 for an interpretation of d.

06 PJN d Plus Jump

Provides a conditional jump of up to 31 steps forward or backward from the current program address if the A register is positive. If A is negative, the next instruction is executed. See instruction 03 for an interpretation of d.

07 MJN d Minus Jump

Provides a conditional jump of up to 31 steps forward or backward from the current program address if the A register is negative. If A is positive, the next instruction is executed. See instruction 03 for an interpretation of d.

10 SHN d Shift

Shifts the contents of the A register right or left d places. If d is positive (00-37$_8$), the shift is left circular; if d is negative (40-77), A is shifted right (end off with no sign extension). Thus, a left shift of 6 places results when d = 6 and a right shift of 6 places results when d = 71.

11 LMN d Logical Difference

Forms in the A register the bit-by-bit logical difference of d and the lower 6 bits of A. This is equivalent to complementing the individual bits in A which correspond to bits in d equal to one. The upper 12 bits of A are not altered.

```
A = 001110101011001001
d =             001010
    ------------------
    001110101011000011
```

12 LPN d Logical Product

Forms in the A register the bit-by-bit logical product of d and the lower 6 bits of A. The upper 12 bits of A are zero.

```
A = 001110101011001001
d =             001010
    ------------------
    000000000000001000
```

13 SCN d Selective Clear

Clears any of the lower 6 bits of the A register where corresponding bits of d are one. The upper 12 bits of A are not altered.

```
A = 001110101011001001
d =             001010
    ------------------
    001110101011000001
```

14 LDN d Load

Clears the A register and loads d into the lower 6 bits of A. The upper 12 bits of A are zero.

15 LCN d Load Complement

Clears the A register and loads the complement of d into the lower 6 bits of A. The upper 12 bits of A are set to one.

16 ADN d Add

Adds the 6-bit positive quantity d to the contents of the A register.

17 SBN d Subtract

Subtracts the 6-bit positive quantity d from the contents of the A register.

20 LDC dm Load

Clears the A register and loads the 18-bit quantity dm, consisting of d as the upper 6 bits and m as the lower 12 bits.

21 ADC dm Add

Adds to the A register the 18-bit quantity dm, consisting of d as the upper 6 bits and m as the lower 12 bits.

22 LPC dm Logical Product

Forms in the A register the bit-by-bit logical product of the contents of A and the 18-bit quantity dm.

```
 A = 001110101011001001
dm = 001110000011001010
     ------------------
     001110000011001000
```

23 LMC dm Logical Difference

Forms in the A register the bit-by-bit logical difference of the contents of A and the 18-bit quantity dm. This is equivalent to complementing the individual bits in A which correspond to bits in dm equal to one.

```
 A = 001110101011001001
dm = 000010000000001010
     ------------------
     001100101011000011
```

24 PSN Pass

A no operation instruction.

25 PSN Pass

A no operation instruction.

26 EXN Exchange Jump

Transmits an 18-bit address from the A register to the central processor with a signal which tells the central processor to perform an exchange jump, with the address in A as the starting location of a file of 16 words containing information about the CP program to be executed. The 18-bit initial address must be entered in A before this instruction is executed. The central processor replaces the file with similar information from the interrupted CP program. The PP program is not interrupted.

27 RPN Read Program Address

Transfers the content of the central processor program address register to the peripheral processor A register to allow the PP to determine whether the central processor is running.

30 LDD (d) Load

Clears the A register and loads the contents of location d into the lower 12 bits of A. The upper six bits of A are zero.

31 ADD (d) Add

Adds to the A register the 12-bit positive quantity contained in location d.

32 SBD (d) Subtract

Subtracts from the A register the 12-bit positive quantity contained in location d.

33 LMD (d) Logical Difference

Forms in the A register the bit-by-bit logical difference of the lower 12 bits of A and the contents of location d. This is equivalent to complementing individual bits of A which correspond to one bits in the contents of location d. The upper 6 bits of A are not altered.

```
  A = 001110101011001001
(d) =       010100001010
      ------------------
      001110111111000011
```

34 STD (d) Store

Stores the lower 12 bits of the A register into location d. The contents of A are not altered.

35 RAD (d) Replace Add

Adds the 12-bit quantity in location d to the contents of the A register and stores the lower 12 bits of the result back in location d. The result is also left in the A register at the end of the operation.

36 AOD (d) Replace Add One

Adds one to the original value in location d and stores the result back in location d. The result is also left in the A register at the end of the operation.

37 SOD (d) Replace Subtract One

Subtracts one from the original value in location d and stores the result back in location d. The result is also left in the A register at the end of the operation.

40 LDI ((d)) Load

Clears the A register and loads into A the 12-bit quantity obtained by indirect addressing. The upper 6 bits of A are zero.

41 ADI ((d)) Add

Adds to the contents of the A register a 12-bit positive operand obtained by indirect addressing.

42 SBI ((d)) Subtract

Subtracts from the A register a 12-bit positive operand obtained by indirect addressing.

43 LMI ((d)) Logical Difference

Forms in the A register the bit-by-bit logical difference of the lower 12-bits of A and the 12-bit operand obtained by indirect addressing. This is equivalent to complementing individual bits of A which correspond to one bits in the operand. The upper 6 bits of A are not altered.

```
    A = 001110101011001001
((d)) =       010100001010
        ------------------
        001110111111000011
```

44 STI ((d)) Store

Stores the lower 12 bits of the A register into the location specified by the contents of location d. The contents of A are not altered.

45 RAI ((d)) Replace Add

Adds to the contents of the A register the operand obtained from the location specified by the contents of location d. The resultant sum is left in the A register at the end of the operation and the lower 12 bits of A replace the original operand in memory.

46 AOI ((d)) Replace Add One

Adds one to the operand obtained from the location specified by the contents of location d. The resultant sum is left in the A register at the end of the operation and the lower 12 bits of A replace the original operand in memory.

47 SOI ((d)) Replace Subtract One

Subtracts one from the operand obtained from the location specified by the contents of location d. The resultant difference is left in the A register at the end of the operation and the lower 12 bits of A replaces the original operand in memory.

50 LDM (m + (d)) Load

Clears the A register and loads a 12-bit operand obtained by indexed direct addressing into the lower 12 bits of A. The upper 6 bits of A are zero. If d = 0, the operand address is simply m. If d ≠ 0, then m plus the contents of location d is the operand address. Thus the contents of d may be used as an index quantity to modify operand addresses.

51 ADM (m + (d)) Add

Adds to the contents of the A register a 12-bit positive operand obtained by indexed direct addressing. (See instruction 50 for further explanation of addressing.)

52 SBM (m + (d)) Subtract

Subtracts from the A register a 12-bit positive operand obtained by indexed direct addressing. (See instruction 50 for further explanation of addressing.)

53 LMM (m + (d)) Logical Difference

Forms in the A register the bit-by-bit logical difference of the lower 12 bits of A and a 12-bit operand obtained by indexed direct addressing. This is equivalent to complementing individual bits of A which correspond to one bits in the operand. The upper 6 bits of A are not altered.

$$\begin{array}{rr} A = & 001110101011001001 \\ (m + (d)) = & 010100001010 \\ \hline & 001110111111000011 \end{array}$$

54 STM (m + (d)) Store

Stores the lower 12 bits of the A register in the location determined by indexed direct addressing. The contents of A are not altered. (See instruction 50 for further explanation of addressing.)

55 RAM (m + (d)) Replace Add

Adds the contents of the A register to the operand obtained from the location determined by indexed direct addressing. The resultant sum is left in the A register at the end of the operation and the lower 12 bits of A replace the original operand in memory. (See instruction 50 for further explanation of addressing.)

56 AOM (m + (d)) Replace Add One

Adds one to the operand obtained from the location determined by indexed direct addressing. The resultant sum is left in the A register at the end of the operation and the lower 12 bits of A replace the original operand in memory. (See instruction 50 for further explanation of addressing.)

57 SOM (m + (d)) Replace Subtract One

Subtracts one from the operand obtained from the location determined by indexed direct addressing. The resultant difference is left in the A register at the end of the operation and the lower 12 bits of A replace the original operand in memory. (See instruction 50 for further explanation of addressing.)

60 CRD d Central Read from (A) to d

Transfers a 60-bit central memory word to 5 consecutive PP memory locations. The A register must contain the 18-bit CM address before the instruction is executed. The 60-bit CM word is disassembled beginning at the left, with the location specified by d receiving the left most 12-bit word; d + 1, the next 12-bit word, and so on.

61 CRM m d Central Read (d) words from (A) to m

Reads a block of 60-bit words from central memory into peripheral processor memory. The A register contains the 18-bit CM starting address and must be loaded prior to the execution of this instruction. The contents of A are increased by one as each 60-bit CM word is disassembled and stored. The block length or number of CM words to be read is contained in location d. The number also goes to the Q register where it is reduced by one as each CM word is processed. The transfer is completed when Q = 0.

The current contents of the P register are stored in PP location 0000, and the PP starting address m goes to the P register. The contents of the P register, m, are increased by one as each 12-bit PP word is stored. The number of PP words required is five times the number of CM words read, since each CM word is disassembled into five successive PP words. The original contents of P are restored upon completion of the transfer.

62 CWD d Central Write from d to (A)

Assembles five successive 12-bit words into a 60-bit word and stores the word in central memory. The 18-bit CM address must be in the A register prior to the execution of the instruction.

The first word to be read out of PP memory is contained in location d. This word appears as the leftmost 12 bits of the 60-bit word. The remaining 12-bit groups are taken from successive addresses in PP memory.

63 CWM m d Central Write (d) words from m to (A)

Assembles a block of 60-bit words and writes them in central memory. The A register contains the beginning central memory address and must be loaded prior to the execution of this instruction. The number in A is increased by one after each 60-bit word is assembled to provide the next CM address.

The contents of location d specify the number of 60-bit words to write. The number also goes to the Q register where it is reduced by one as each CM word is assembled. The transfer is completed when Q = 0.

During execution of this instruction, the original contents of the P register are stored in PP location 0000 and the address of the first word to read from PP memory, m, goes to the P register. The contents of the P register are increased by one as each 12-bit word is read to provide the next PP memory address. The original contents of the P register are restored at the completion of the transfer.

64 AJM m d Jump to m if channel d active

Provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is active. If the channel is inactive, the current program sequence continues.

65 IJM m d Jump to m if channel d inactive

Provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is inactive. If the channel is active, the current program sequence continues.

66 FJM m d Jump to m if channel d full

Provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is full. If the channel is empty, the current program sequence continues.

An input channel is full when the input equipment sends a word to the channel register and sets the full flag. The channel remains full until the PP accepts the word and clears the full flag.

An output channel is full when the PP places a word in the channel register and sets the full flag. The channel is empty when the output equipment accepts the word and notifies the PP.

67 EJM m d Jump to m if channel d empty

Provides a conditional jump to a new program sequence beginning at address m if the channel specified by d is empty. If the channel is full, the current program sequence continues.

70 IAN d Input to A from channel d

Transfers a word from input channel d to the lower 12 bits of the A register.

71 IAM m d Input (A) words from channel d to m

Transfers a block of words from input channel d to PP memory beginning at a location specified by m. The A register contains the block length and is reduced by one as each word is read. The input operation is completed when A = 0.

During this instruction the current contents of the P register are stored in PP location 0000. The P register now holds m and is increased by one as each word is stored to give the address for the next word. The original contents of the P register are restored at the end of the operation.

72 OAN d Output (A) on channel d

Transfers a word from the lower 12 bits of the A register to output channel d.

73 OAM m d Output (A) words from m on channel d

Transfers a block of words on output channel d from PP memory beginning at the location specified by m. The contents of the A register specify the number of words to be sent out and are reduced by one as each word is sent. The output operation is completed when A = 0.

During this instruction the current contents of the P register are stored in PP location 0000. The P register now holds m and is increased by one as each word is read out to give the address of the next word. The original contents of the P register are restored at the end of the operation.

74 ACN d Activate channel d

Activates the channel specified by d. This instruction must precede a 70-73 instruction. Activating a channel alerts and prepares the I/O equipment for the exchange of data.

75 DCN d Disconnect channel d

Deactivates the channel specified by d. This stops the I/O equipment and the buffer terminates.

76 FAN d Function (A) on channel d

Sends out on channel d the external function code in the lower 12 bits of the A register.

77 FNC m d Function m on channel d

Sends out on channel d the external function code specified by m.

4. PSEUDO OPERATION CODES

Pseudo operations provide the means for directing the assembler to carry out certain functions. The instruction format is the same as the basic format shown in 2.1. As used here, LOC indicates that the particular operation may have a symbolic identifier in the location field. Where none is shown, those columns are ignored by the assembler. Following are ASPER pseudo operations and meanings.

ASPER P1

Causes the compiling process to enter peripheral processor assembly mode. P1 names the peripheral processor program and is referenced by the TPP macro. Each peripheral processor program associated with the same central processor program must have a unique name, P1.

SUPB P1, P2

Defines the coding that follows to be a subprogram or overlay. P1 names the subprogram and is referenced by the LOAD macro. Each subprogram of all peripheral processor programs associated with the same central processor program must have a unique name, P1. P2 is the peripheral processor location at which the segment is to be loaded. P2 may be any of the legal operands, excluding central memory symbols. Any PP symbols appearing in P2 must be previously defined.

ORG

Assigns the coding that follows to the direct area of peripheral processor memory and prohibits its relocation at load time. The ORG pseudo allows the programmer to define temporary and index locations within the first 64 core locations by setting the location counter to address 0010 and labeling the following coding and symbols as non-relocatable. The setting of the location counter to 0010 omits the assignment of locations 0000 to 0007 because these locations have unique significance to the peripheral processor and the SIPROS resident and generally should not be used. The assembler does not sense the end of the direct area but allows the location counter to increment through 77_8.

ORGR

Assigns the coding that follows to the non-direct area of peripheral processor memory and labels that coding as relocatable. The location counter is set to memory location 0100_8 and is allowed to increment through 6777_8. The SIPROS resident exists at execution time at locations 7000_8 through 7776_8. If during assembly the location counter exceds 6777_8, subsequent lines on the output listing are flagged as errors.

BSSD N, L, NAME, R

Defines on logical disk unit N the file identified by NAME which has L number of 60-bit words in its longest record. R specifies the maximum number of logical records into which the file may be segmented. The parameters N, L, and R must be numbers, where $N \leq 16_{10}$, $L \leq 2^{17}$, and $R \leq 4000_{10}$. NAME must be unique within the routine.

LOC BSS CONSTANT

Reserves the number of 12-bit locations specified by CONSTANT beginning at LOC. The contents of the locations reserved are not set to a particular condition. The LOC symbol is equated to the address of the first word of the area. Any symbolic constant appearing in the address field must be previously defined.

LOC BSSZ CONSTANT

Same as BSS except the contents of the locations reserved are set to zero in the object code.

LOC BSSCM CONSTANT

Reserves the number of 60-bit central memory locations specified by CONSTANT. The LOC symbol is equated to the address of the first word of the area in central memory. Any symbolic constant appearing in the address field must be previously defined.

LOC EQU OPERAND

The symbol in the LOC field is assigned the value of the address field. Any symbol appearing in the address field must be previously defined.

LOC DPC $*C_1C_2 - - - C_n*$

Converts the characters enclosed by the asterisks to display code, two characters per word beginning at LOC. Incomplete words are padded out with DPC blanks. The LOC symbol is equated to the address of the first word of the area.

LOC DPC $nnC_1C_2 - - - C_{nn}$

Converts the nn characters, C_1, C_2, - - - C_{nn}, to display code, two characters per word beginning at LOC. The number of characters, nn, must be a two-digit decimal number. Incomplete words are padded out with DPC blanks. The LOC symbol is equated to the address of the first word of the area.

LOC BCD $*C_1C_2 - - - C_n*$

Converts the characters enclosed by the asterisks to BCD code, two characters per word beginning at LOC. Incomplete words are padded out with BCD blanks. The LOC symbol is equated to the address of the first word of the area.

LOC BCD $nnC_1C_2- - - - C_{nn}$

Converts the nn characters C_1, C_2, - - - C_{nn}, to BCD code, two characters per word beginning at LOC. The number of characters, nn, must be a two-digit decimal number. Incomplete words are padded out with BCD blanks. The LOC symbol is equated to the address of the first word of the area.

LOC CON V_o, - - -, V_n

Converts each V_i term to a 12-bit constant. If more than one is defined per a CON pseudo code, each V_i must be separated by one or more spaces, commas, or equal signs. The V_i may be:

a. ± octal integer

b. ± decimal integer

c. symbol

d. symbol±interger

e. symbol−symbol

The LOC symbol is equated to the address of V_o.

LOC COND V_o, - - -, V_n

Same as CON, except each V_i term occupies two words where the V_i term is converted to 18 bits with the most significant 6 bits right justified in the first word and the remaining 12 bits in the second word.

END

Terminates the assembly process for this job. When punched into a card, END must be the only entry in the card and must start in column 11.

LIST P

Controls the listing of the side-by-side, so that sections of coding may be omitted from the listing. At the beginning of each ASPER program the assembler assumes the list case of P = 0, unless otherwise specified by the LIST pseudo opcode.

If P = 0, list the side-by-side that follows.

If P ≠ 0, suppress the side-by-side listing.

SPACE nn

Spaces nn lines on the listing. The decimal number, nn, is evaluated $\leqq 63_{10}$.

EJECT

Ejects listing to the top of the next page.

5. SYSTEM MACROS

System macro instructions provide communication links between an ASPER routine in a peripheral processor and the system peripheral processors. While most of these macros direct the operating system to perform input/output operations, others request equipment assignment, check the status of external operations, produce program overlays, utilize system peripheral processors in conjunction with the ASPER peripheral processor, and request the operating system to provide channel scheduling services.

The communication link provided by the system macros allows a two-way information transfer. The ASPER routine not only gives the system peripheral processor request information but also a location in central memory in which the system peripheral processor enters the status of the requested operation, reporting its success back to the ASPER routine. Each system macro must have a status response word, which is set by the operating system in performing the function of the individual macro request.

All communication links are made through central memory. The status response word identified by the request must be a central memory word. Similarly, requests for the system to input or output data for an ASPER routine assume the data region is located in central memory. The regions required may be defined either by the central processor program, if one is used, or by the ASPER BSSCM pseudo operation.

Where applicable, system macros provide a buffered and a non-buffered mode. In the buffered mode, the macro used without an appended "W," the ASPER routine is free to continue processing while waiting for the results requested. However, it must do its own status checking, by means of another macro, to determine when the requested operation is completed. In the non-buffered mode, with the "W" appended to the macro opcode, the macro itself determines when the requested operation is completed or aborted, and PP processing is discontinued until the results are known. Both modes return full information on the status of the request.

For each system macro encountered, ASPER generates a sequence of coding which communicates the requested function to the system through the peripheral processor resident program. The coding consists of a Return Jump to the resident routine followed by an Unconditional Jump past a vector of words containing the octal opcode, buffer-mode flag, and evaluated parameters.

A list of required parameters is specified for each system macro. These parameters may be written in various forms depending on the type of parameter. Parameters representing peripheral processor locations which contain the actual parameter may be written in the forms:

SYMBOL

SYMBOL±CONSTANT

Parameters representing central memory locations may be written in the LITERAL forms:

(SYMBOL)

(SYMBOL±CONSTANT)

Parameters representing numbers per se may be written in the forms:

CONSTANT

SYMBOL

SYMBOL±CONSTANT

Parameters representing a file or program name must be written in the form:

SYMBOL

The following is an explanation of certain letters, terms, and phrases which are used frequently in connection with macros.

A	Symbolic address in PP memory which contains the CM address of the first word of the requested block assigned by the system or which contains the CM address, as specified by the programmer, of the first word of the block in memory to be released to the system. If the macro is used in a PP program, the CM address is absolute; but if used in a CP program, the address is relative.
BA	Symbolic address in PP memory which contains the beginning address of the buffer area in central memory.

C Conversion mode

Card operations:

C = blank or 0 – no conversion (binary image)

1 – Hollerith to display code for read; display code to Hollerith for punch

2 – Hollerith to BCD for read; BCD to Hollerith for punch

Magnetic tape:

C = blank or 0 – no conversion

1 – BCD to display code

2 – display code to BCD

Printer:

C = blank or 0 – no conversion

2 – display code to BCD

D Physical number (or PP symbolic address of number) of the I/O channel requested or released.

EA Symbolic address in PP memory which contains the ending address + 1 of the buffer area in central memory.

K Number (or PP symbolic address of number) of logical tape records.

L Number (or PP symbolic address of number) of 60-bit words in the longest record in the file identified by NAME.

N Equipment logical number (or PP symbolic address of number), i.e., 1, 2, . . . M for M total units of equipment type in the system.

NAME Symbolic name uniquely identifying the disk logical file being referenced.

NW Total number (or PP symbolic address of number) of central memory words requested or released.

P Logical record number (or PP symbolic address of number) in disk file to start read or write.

R Maximum number (or PP symbolic address of number) of logical records into which the disk file may be segmented.

RL Record length

Card operations: number (or PP symbolic address of number) of leftmost 5 columns (binary image) or 10-character fields (coded mode) of the card. For BCD or DPC conversion mode, each 60-bit word contains ten 6-bit characters. For binary image, each 60-bit word contains 5 columns.

Console operations: total number (or PP symbolic address of number) of characters in the message to be transmitted.

Magnetic tape: number (or PP symbolic address of number) of 60-bit words per tape record.

Printer: number (or PP symbolic address of number) of 10-character words per line to print.

S Symbolic address in PP memory which contains the CM address for the STATUS RESPONSE WORD from the PP I/O routine. A peripheral processor program requires that a location in central memory be reserved and identified for each macro request. The system enters the status of the requested operation in this location.

SYMBOL Symbolic Name

Program overlay: name of overlay region to be loaded.

System action: name of PP program defined by ASPER pseudo operation.

Wait check: name of transfer location if abort is indicated by the status response word.

T Display character size:

T = blank or 0 – 64 characters/line
1 – 32 characters/line
2 – 16 characters/line
3 – plot mode

TAG Identification number $\leq$ 18 bits (or PP symbolic address of number) of message to be displayed.

W A W appended to the opcode of a macro indicates a "wait for reply." If the W is not used (buffered mode), the routine may continue processing while the requested I/O operation is being performed. However, the routine must do its own checking on the progress of the request by means of the WAI (Wait Check) macro. If the request is in process, the status response word is positive and nonzero; if the request is completed, the word is zero; if the request is aborted, the word is negative.

When the W is appended to the macro (nonbuffered mode) and the requested operation can be performed, the routine delays until the status response word is zero (completed) or negative (aborted).

In both modes if the requested operation is successful, the next in-line instruction is executed.

5.1 MAGNETIC TAPE OPERATIONS

OPCODE	ADDRESS FIELD	REMARKS	
RQTW	N, S	Request tape assignment from system.	Wait if W used.
DRTW	N, S	Release tape back to system.	Wait if W used.
SFFW	N, S	Search file mark forward.	Wait if W used.
SFBW	N, S	Search file mark backward.	Wait if W used.
WFMW	N, S	Write file mark.	Wait if W used.
RWLW	N, S	Rewind tape to load point.	Wait if W used.
RWUW	N, S	Rewind tape for unload.	Wait if W used.
FSPW	N, S, K	Forespace	Wait if W used.
BSPW	N, S, K	Backspace	Wait if W used.
RFCW	N, S, BA, EA, RL, C	Read tape forward coded mode.	Wait if W used.
RFBW	N, S, BA, EA, RL, C	Read tape forward binary mode.	Wait if W used.
WRCW	N, S, BA, EA, RL, C	Write tape coded mode.	Wait if W used.
WRBW	N, S, BA, EA, RL, C	Write tape binary mode.	Wait if W used.

N = Magnetic tape logical unit number; 1, 2, . . . M for M tape units in the system.

S = Location containing the central memory address for status response code from System PP I/O routine.

K = Number of logical tape records.

BA = Location containing the beginning address of buffer area in central memory.

EA = Location containing the ending address + 1 of buffer area in central memory.

RL = Number of 60-bit words per tape record.

C = Conversion mode.

Blank or 0 — No conversion.

1 — BCD to Display Code.

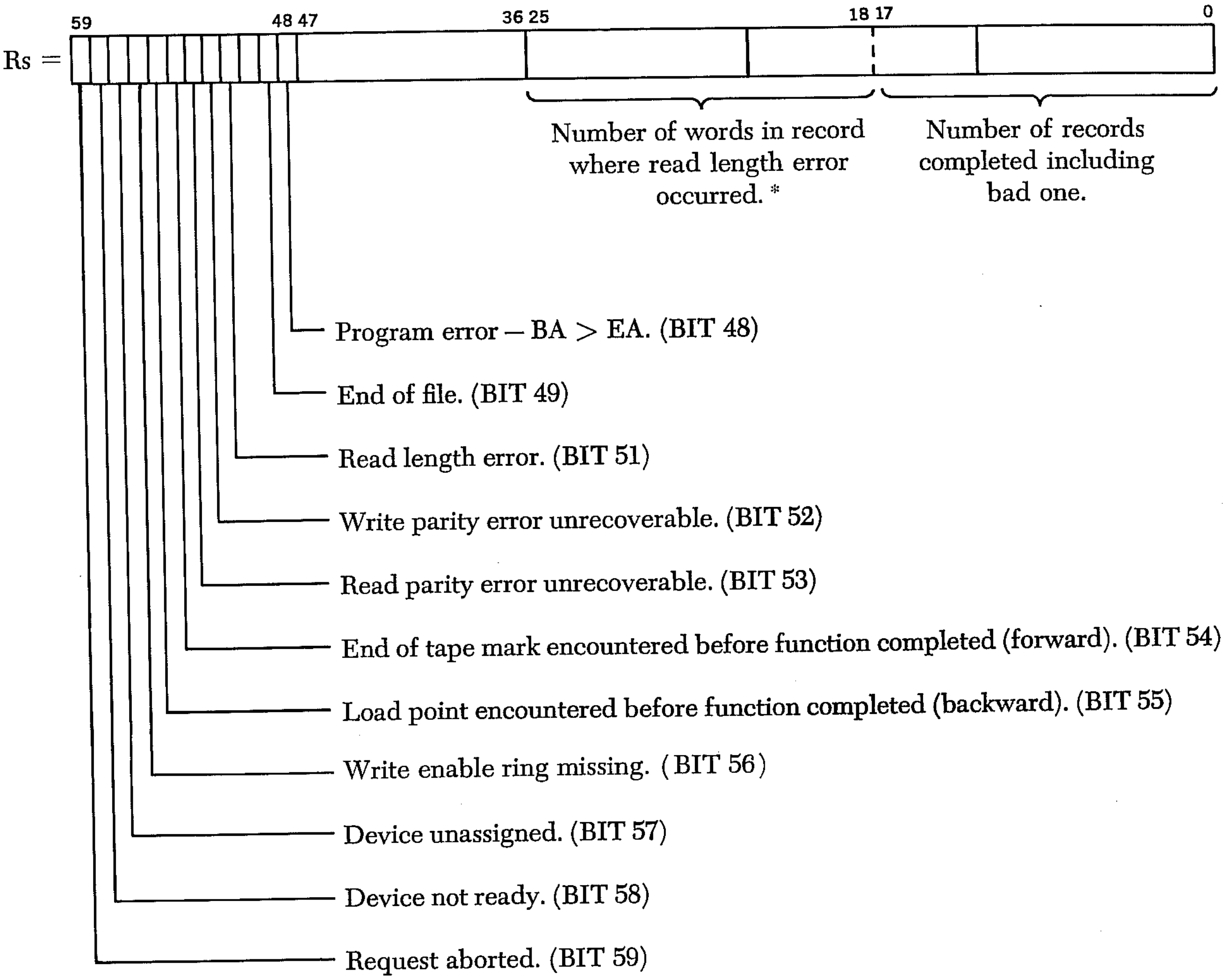
2 — Display Code to BCD.

STATUS RESPONSE WORD — positioned as per address S.

Rs = 0 Request completed with no trouble.

Rs = 1 Request in process.

Rs < 0 Request aborted. Reason give in bits 58-48.


Rs =
59
48 47
36 25
18 17
0
Number of words in record where read length error occurred. *
Number of records completed including bad one.
Program error — BA > EA. (BIT 48)
End of file. (BIT 49)
Read length error. (BIT 51)
Write parity error unrecoverable. (BIT 52)
Read parity error unrecoverable. (BIT 53)
End of tape mark encountered before function completed (forward). (BIT 54)
Load point encountered before function completed (backward). (BIT 55)
Write enable ring missing. (BIT 56)
Device unassigned. (BIT 57)
Device not ready. (BIT 58)
Request aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

*Refers to peripheral processor words.

5.2 DISK TRANSFERS

Provision is made in the operating system for the programmer to read and write scratch data to and from disk storage units. Data are usually broken up into related blocks called *files*. The files, in turn, are segmented into the blocks of data that are transmitted at one time. These are called *logical records*. For most efficient utilization of disk storage, logical records contain a minimum of 512 central memory words. A file is defined by the ASPER pseudo operation, BSSD, which specifies the number of 60-bit words in the longest record, the maximum number of logical records into which the file is to be segmented, and the symbolic name by which to identify the file. The actual data transmission is accomplished through the use of the following macro operators.

OPCODE	ADDRESS FIELD	REMARKS	
RDHW	N, S, BA, EA, NAME, P	Read record and hold data on disk.	Wait if W used.
RDRW	N, S, BA, EA, NAME, P	Read record and release data on disk.	Wait if W used.
WRDW	N, S, BA, EA, NAME, P	Write record on disk.	Wait if W used.

N = Disk logical unit number; 1, 2, . . . M for M disk units in the system.

S = Location containing the central memory address for status response code from System PP I/O routine.

BA = Location containing the beginning address of buffer area in central memory.

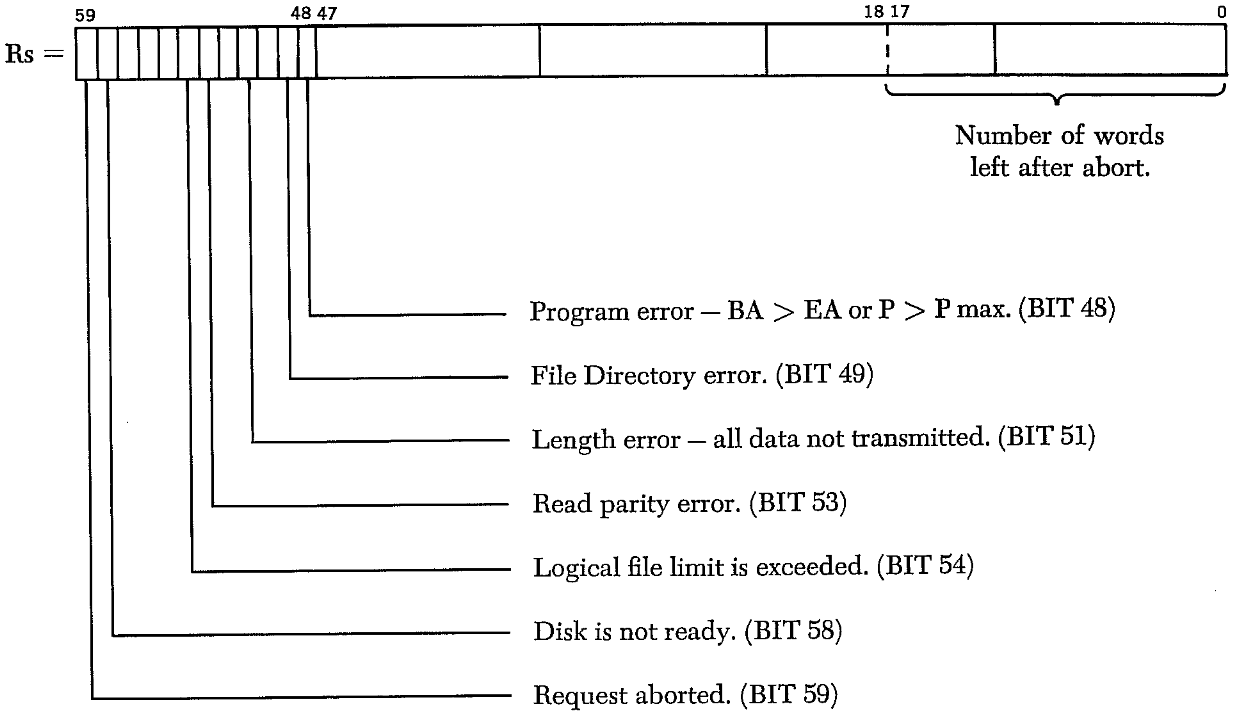
EA = Location containing the ending address + 1 of buffer area in central memory.

NAME = Symbolic name to identify disk logical file to be referenced.

P = Logical record number used to identify record read from disk or written onto disk.

STATUS RESPONSE WORD – positioned as per address S.

Rs = 0	Request is completed with no trouble.
Rs = 1	Request is in process.
Rs < 0	Request aborted. Reason given in bits 58-48.



Number of words left after abort.

Program error — BA > EA or P > P max. (BIT 48)

File Directory error. (BIT 49)

Length error — all data not transmitted. (BIT 51)

Read parity error. (BIT 53)

Logical file limit is exceeded. (BIT 54)

Disk is not ready. (BIT 58)

Request aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

5.3 PRINTER OPERATIONS

OPCODE	ADDRESS FIELD	REMARKS	
SSPW	N, S	Single space printer.	Wait if W is used.
DSPW	N, S	Double space printer.	Wait if W is used.
FC7W	N, S	Select Format Channel 7.	Wait if W is used.
FC8W	N, S	Select Format Channel 8.	Wait if W is used.
MC1W	N, S	Select Monitor Channel 1.	Wait if W is used.
MC2W	N, S	Select Monitor Channel 2.	Wait if W is used.
MC3W	N, S	Select Monitor Channel 3.	Wait if W is used.
MC4W	N, S	Select Monitor Channel 4.	Wait if W is used.
MC5W	N, S	Select Monitor Channel 5.	Wait if W is used.
MC6W	N, S	Select Monitor Channel 6.	Wait if W is used.
CMCW	N, S	Clear Monitor Channels 1 - 6.	Wait if W is used.
SPAW	N, S	Suppress space after next print.	Wait if W is used.
PRNW	N, S, BA, EA, RL, C	Print single line or multiple lines.*	Wait if W is used.

*If SPA is given preceding a multiple line print, it applies only to the first line.

N = Printer logical unit number; 1, 2, . . . M for M printers in the system.

S = Location containing the central memory address for status response code from System PP I/O routine.

BA = Location containing the beginning address of buffer area in central memory.

EA = Location containing the ending address + 1 of buffer area in central memory.

RL = Number of 10 character words per line to print.

C = Conversion mode.

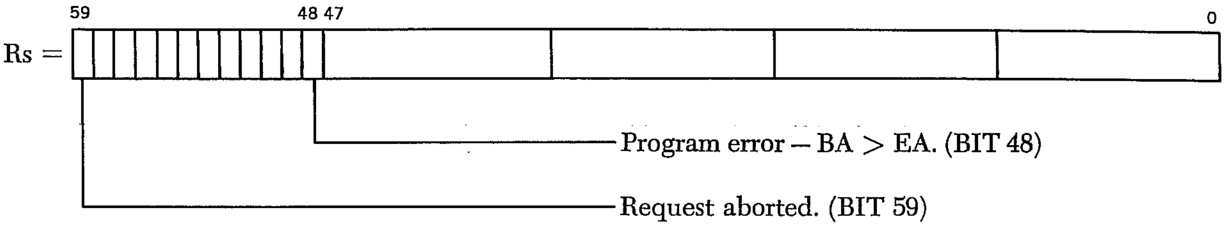
Blank or 0 — No conversion.

2 — Display Code to BCD.

Printer character codes are given in Table 4 of the Appendix.

STATUS RESPONSE WORD — positioned as per address S.

Rs = 0	Request is completed with no trouble.
Rs = 1	Request is in process.
Rs < 0	Request aborted. Reason given in bits 58-48.



Program error — BA > EA. (BIT 48)

Request aborted. (BIT 59)

where: 1 implies the condition exists.

0 implies the condition does not exist.

5.4 CARD OPERATIONS

OPCODE	ADDRESS FIELD	REMARKS	
PCHW	N, S, BA, EA, RL, C	Punch cards.	Wait if W is used.
RDCW	N, S, BA, EA, RL, C	Read cards.	Wait if W is used.

N = Card reader or punch logical unit number; 1,2, . . . M for M readers or punches in the system.

S = Location containing the central memory address for status response code from System PP I/O routine.

BA = Location containing the beginning address of buffer area in central memory.

EA = Location containing the ending address + 1 of buffer area in central memory.

RL = Number of leftmost 10-character fields or 5 columns of the card.

C = Conversion mode.

Blank or 0 — No conversion; i.e., binary image input/output.

1 — Hollerith to Display Code for read; Display Code to Hollerith for punch.

2 — Hollerith to BCD for read; BCD to Hollerith for punch.

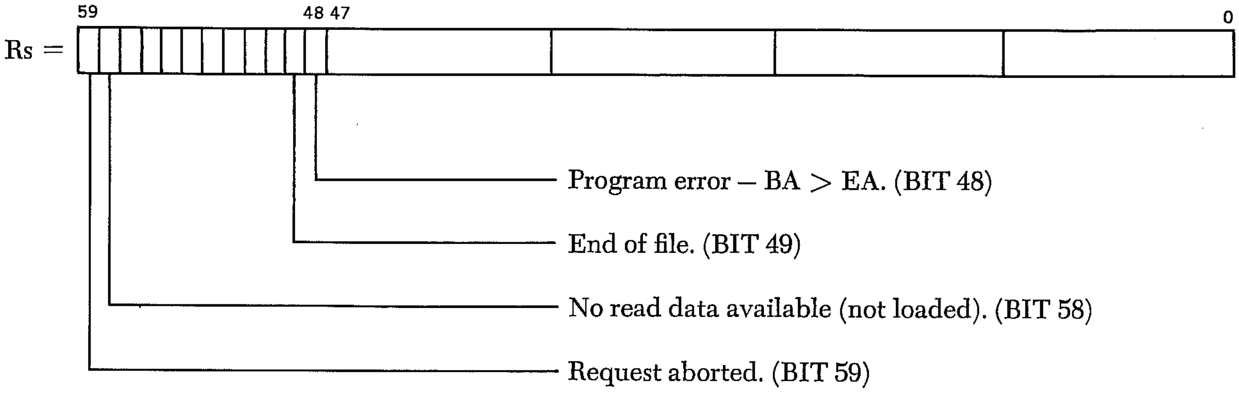
Display character codes are given in Table 4 of the Appendix.

STATUS RESPONSE WORD — positioned as per address S.

Rs = 0 Request is completed with no trouble.

Rs = 1 Request is in process.

Rs < 0 Request aborted. Reason given in bits 58-48.



Program error — BA > EA. (BIT 48)

End of file. (BIT 49)

No read data available (not loaded). (BIT 58)

Request aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

5.5 CONSOLE OPERATIONS

Request procedures are provided for ASPER routines to display messages on the primary console right scope or either of the scopes on other consoles. The system provides a timing service for removal of displays after a certain exposure. However, the request procedure gives an option to override the system time limit on display. In this mode, it is assumed that ASPER routine will request a removal of the display as a result of console acknowledgment or internal decision.

OPCODE	ADDRESS FIELD	REMARKS
DSRW	N, S, BA, EA, RL, TAG, T	Display on Right Scope for system time limit. Wait if W is used.
DSLW	N, S, BA, EA, RL, TAG, T	Display on Left Scope for system time limit. Wait if W is used.
DHRW	N, S, BA, EA, RL, TAG, T	Display on Right Scope and hold indefinitely. Wait if W is used.
DHLW	N, S, BA, EA, RL, TAG, T	Display on Left Scope and hold indefinitely. Wait if W is used.
RDPW	N, S, TAG	Remove display. Wait if W is used.
RTYW	N, S, BA, EA, RL, TAG	Read console typewriter. Wait if W is used.

N = Console logical unit number; 1, 2, . . . M for M consoles in the system.

S = Location containing the central memory address for status response code from System PP I/O routine.

BA = Location containing the beginning address of buffer area in central memory.

EA = Location containing the ending address + 1 of buffer area in central memory.

RL = Total number of characters in the message to be transmitted.

TAG = Identification number $\leq$ 18 bits for display message.

T = Display character size.

Blank or 0 – 64 characters/line.

1 – 32 characters/line.

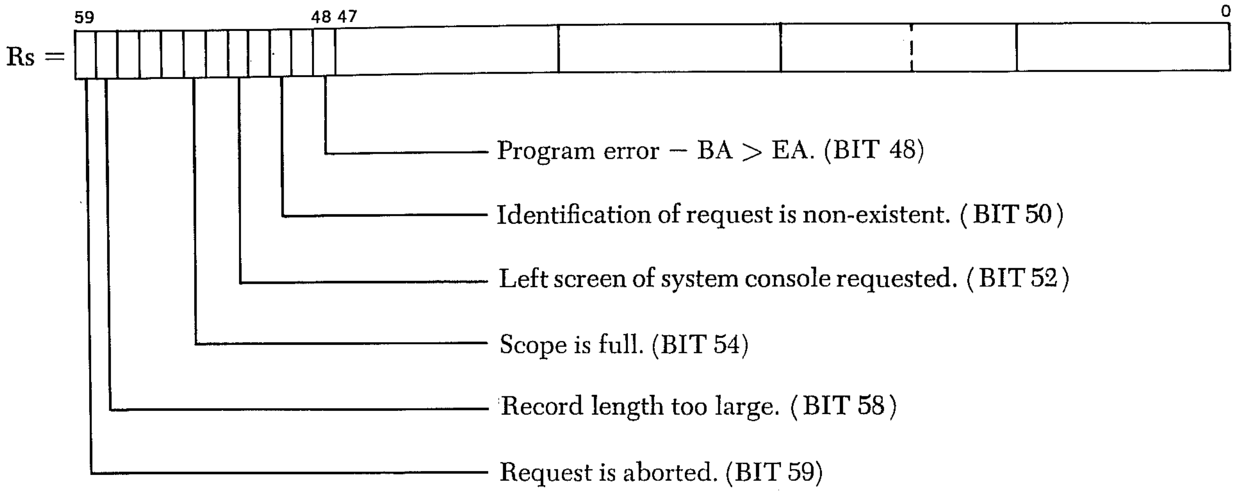
2 – 16 characters/line.

3 – plot mode.

Display character codes are given in Table 4 of the Appendix.

STATUS RESPONSE WORD – positioned as per address S.

Rs = 0	Request is completed with no trouble.
Rs = 1	Request is in process.
Rs < 0	Request aborted. Reason given in bits 58-48.



Program error – BA > EA. (BIT 48)

Identification of request is non-existent. (BIT 50)

Left screen of system console requested. (BIT 52)

Scope is full. (BIT 54)

Record length too large. (BIT 58)

Request is aborted. (BIT 59)

where: 1 implies the condition exists.
0 implies the condition does not exist.

5.6 SYSTEM ACTION

OPCODE	ADDRESS FIELD	REMARKS
TPPW	N, S, SYMBOL	Transfer program SYMBOL from CM to PP memory and begin execution with first ASPER instruction. Wait if W is used.
RQMW	NW, S, A	Request memory. Wait if W is used.
DRMW	NW, S, A	Release memory. Wait if W is used.
RQDW	N, S, L, NAME, R	Request disk space. Wait if W is used.
DRDW	N, S, NAME	Release disk space. Wait if W is used.
RQCW	D, S	Request I/O channel. Wait if W is used.
DRCW	D, S	Request I/O channel. Wait if W is used.
DRPP	N, S	Release peripheral processor.

D = Physical number of I/O channel.

N = Logical number of PP or disk unit.

S = Location containing the central memory address for status response code from System PP I/O routine.

R = Maximum number of logical records into which the file may be segmented.

NW = Total number of words.

L = Number of 60-bit words in longest record.

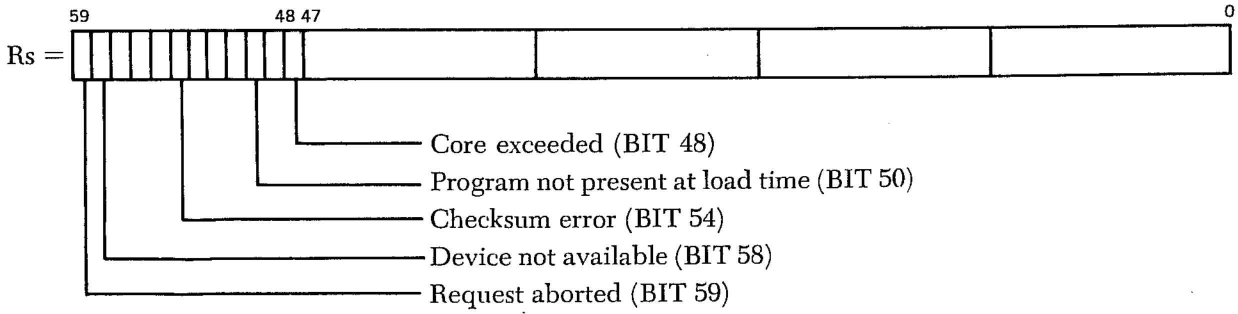
A = Location containing the central memory address of the first word of block assigned by the system or released by the programmer.

NAME = Symbolic name uniquely identifying the disk logical file being referenced.

SYMBOL = Name of PP program defined by ASPER pseudo operation.

STATUS RESPONSE WORD – positioned as per address S.

Rs = 0	Request completed with no trouble.
Rs = 1	Request in process.
Rs < 0	Request aborted. Reason given in bits 58-48.



where: 1 implies the condition exists.
0 implies the condition does not exist.

5.7 PROGRAM OVERLAY

No execution takes place unless all SUBP's called in LOAD macros are present. During execution of the LOAD macro, control is kept in the macro and returned to the routine only upon successful completion of the load. Therefore, no status is provided. A further explanation of program segmentation and the overlay operation is given in Section 8.

OPCODE	ADDRESS FIELD	REMARKS
LOAD	SYMBOL	Load SUBP SYMBOL into PP memory.

SYMBOL = Name of overlay region to be loaded.

5.8 WAIT CHECK

After a buffered operation is initiated, a Wait Check macro may be used to check status. The routine delays until the status response word is zero (completed) or negative (aborted). If it is zero, the next instruction in line is executed. If the status word is negative, the routine exits to the location specified by SYMBOL.

OPCODE	ADDRESS FIELD	REMARKS
WAIW	S, SYMBOL	Check status of S. Exit to SYMBOL if abort. Wait for reply if not ready and W is used.

S = Location containing the central memory addrerr for status response code from System PP I/O routine.

SYMBOL = Transfer location if an abort is indicated by the status response code.

6. MACRO INSTRUCTIONS

6.1 DESCRIPTION

Backspace

BSP N, S, K

Backspaces K number of records on logical tape unit N.

Clear Monitor Channels 1-6

CMC N, S

Deselects monitor channels 1-6 on line printer N. This macro must be used before selecting another channel.

Display on Left Scope and Hold Indefinitely

DHL N, S, BA, EA, RL, TAG, T

Displays a message on the left scope of the console and holds the display indefinitely or until an RDP request is received. When displayed the message is accompanied by the 18-bit identifier, TAG. BA and EA contain the locations for the beginning and ending addresses of the buffer area storing the message to be displayed. Each CM word contains 10 consecutive display-coded characters of the message ordered from left to right in the word. The display character size is determined by T. RL specifies the number of characters to be displayed on each line on the scope and is limited by the character size chosen. The logical console number, N, indicates which console is to be used. See Example 1.

Display on Right Scope and Hold Indefinitely

DHR N, S, BA, EA, RL, TAG, T

Displays a message on the right scope of the console and holds the display indefinitely or until an RDP request is received. See macro DHL for further explanation of parameters.

Release Channel Back to System

DRC D, S

Releases the channel specified by D back to the system for general purpose use.

Release Disk Space Back to System

DRD N, S, NAME

Releases the file indentified by NAME on the logical disk unit N.

Release Memory

DRM NW, S, A

Releases from the block of central memory words which the PP has reserved the total number of words specified by NW beginning with the CM address given in A.

Release Tape Back to System

DRT N, S

Releases the logical tape unit specified by N for general system usage.

Display on Left Scope for System Time Limit

DSL N, S, BA, EA, RL, TAG, T

Displays a message on the left scope of the console for the length of time set by the system. See macro DHL for further explanation of parameters.

Double Space Printer

DSP N, S

Advances logical printer N two lines.

Display on Right Scope for System Time Limit

DSR N, S, BA, EA, RL, TAG, T

Displays a message on the right scope of the console for the length of time set by the system. See macro DHL for further explanation of parameters.

Select Format Channel 7

FC7 N, S

Selects format channel 7 on logical printer unit N. This format channel advances the paper to a selected line.

Select Format Channel 8

FC8 N, S

Selects format channel 8 on logical printer unit N. This format channel ejects the page to the top of the form.

Forespace

FSP N, S, K

Spaces forward K number of records on logical tape unit N.

Select Monitor Channel 1

MC1 N, S

Selects monitor channel 1 on logical printer unit N. The monitor channels contain predesigned line-space formats.

Select Monitor Channel 2

MC2 N, S

Select monitor channel 2 on logical printer unit N.

Select Monitor Channel 3

MC3 N, S

Select monitor channel 3 on logical printer unit N.

Select Monitor Channel 4

MC4 N, S

Select monitor channel 4 on logical printer unit N.

Select Monitor Channel 5

MC5 N, S

Select monitor channel 5 on logical printer unit N.

Select Monitor Channel 6

MC6 N, S

Select monitor channel 6 on logical printer unit N.

Punch Cards

PCH N, S, BA, EA, RL, C

Punches cards on logical unit N for the number of leftmost 5 columns (binary output, no conversion) or 10-character fields (coded mode) as given by RL. The conversion mode is specified by C. The card images are read from central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

Print Single Line or Multiple Lines

PRN N, S, BA, EA, RL, C

Prints on logical unit N the number of 10-character words per line as given by RL in the conversion mode specified by C. RL may specify up to 12 or 14* words per line. The print image is stored in central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

Read Card

RDC N, S, BA, EA, RL, C

Reads cards on logical unit N for the number of leftmost 5 columns (binary input, no conversion) or 10-character fields (coded mode) as given by RL. The conversion mode is specified by C. The cards are read into central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

*For the 120 character/line 1612 printer and the 136 character/line 501 printer, respectively.

Read Record and Hold Data on Disk

RDH N, S, BA, EA, NAME, P

Reads into the buffer area in central memory the logical record specified by P of the file identified by NAME from logical disk N. The words are read, without code translation, into the buffer area beginning at the address contained in location BA and ending at the address contained in location EA. The data are held on disk for subsequent re-use.

Remove Display

RDP N, S, TAG

Erases from the scope at console N the display identified by TAG.

Read Record and Release Data on Disk

RDR N, S, BA, EA, NAME, P

Reads into the buffer area in central memory the logical record specified by P of the file identified by NAME from logical disk N. The words are read, without code translation, into the buffer area beginning at the address contained in location BA and ending at the address contained in location EA. Once the data are in memory, the disk space is released for use by other programs.

Read Tape Forward, Binary Mode

RFB N, S, BA, EA, RL, C

Reads, in binary parity, the number of 60-bit words per tape record, RL, from logical tape unit N. Each 6-bit character is converted as specified by the conversion mode C. The words are read into a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

Read Tape Forward, Coded Mode

RFC N, S, BA, EA, RL, C

Reads, in BCD parity, the number of 60-bit words per tape record, RL, from logical tape unit N. Each 6-bit character is converted as specified by the conversion mode C. The words are read into a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

Request Channel

RQC D, S

Requests the channel specified by D for the exclusive use of the requesting PP program.

Request Disk Space

RQD N, S, L, NAME, R

Reserves on logical disk unit N the file identified by NAME which has L number of 60-bit words in its longest record. R specifies the maximum number of logical records into which the file may be segmented. The parameters N, L, and R must be numbers, where $N \leq 16_{10}$, $L \leq 2^{17}$, and $R \leq 4000_{10}$. NAME must be unique within the routine.

Request Memory Space

RQM NW, S, A

Reserves in central memory the total number or words specified by NW. The system sets A to the location containing the address of the first word of the assigned block in central memory.

Request Tape Assignment from System

RQT N, S

Requests logical tape unit N for the exclusive use of a program.

Read Console Typewriter

RTY   N, S, BA, EA, RL, TAG

Reads and identifies a message with the identification number, TAG, typed on the typewriter at logical console unit N. Transmits RL number of characters to a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA.

Rewind Tape to Load Point

RWL   N, S

Rewinds logical tape unit N to the physical load point on the tape.

Rewind Tape for Unload

RWU   N, S

Rewinds logical tape unit N so that the tape may be dismounted.

Search File Mark Backward

SFB   N, S

Searches the tape on logical unit N one record at a time back towards the load point until a file mark is passed over. When the mark is found, the tape is positioned on the load-point side of the file mark. If none is found, the macro is equivalent to RWL.

Search File Mark Forward

SFF   N, S

Searches the tape on logical unit N one record at a time from the current position forward until a file mark is passed over. When the mark is found, the tape is positioned on the side of the file mark away from the load point. If no mark is found, the end of tape marker stops the search.

Suppress Space After Next Print

SPA   N, S

Suppresses on logical printer N the automatic advance after the next line printed with a PRN macro.

Single-Space Printer

SSP   N, S

Advances logical printer N one line.

Transfer PP Program and Begin Execution

TPP   N, S, SYMBOL

Produces a calling sequence to the PP loader which, during execution, transfers PP program SYMBOL from central memory to logical peripheral processor N and begins execution with the first ASPER instruction defined under an ORGR pseudo code. This macro is used to load an ASPER program into a PP from CM at execute time. The load begins at the first binary card and continues until the loader encounters another ASPER header card, a SUBP header card, or a terminate card.

The TPP call from a CM program can load any PP in the system. However, the TPP call by a PP program can load any other PP in the system but cannot load itself.

Wait Check

WAI   S, SYMBOL

Checks the status response word of other macros during a buffered operation. If the operation has been aborted, the WAI macro exits to the address specified by SYMBOL. If not, the next instruction, in line, is executed.

Write File Mark

WFM   N, S

Writes an end of file mark on the tape on logical unit N.

Write Tape, Binary Mode

```
WRB    N, S, BA, EA, RL, C
```

Writes, in binary parity, the data between BA and EA in records of RL 60-bit words each onto logical tape unit N. Each 6-bit character transferred is converted as requested by the conversion mode C. The words are written from a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA. If the conversion mode is 0, a straight binary output is expected. If one of the other conversion modes is used, Example 2 applies.

Write Tape, Coded Mode

```
WRC    N, S, BA, EA, RL, C
```

Writes, in BCD parity, the data between BA and EA in records of RL 60-bit words each onto logical tape unit N. Each 6-bit character transferred is converted as requested by the conversion mode C. The words are written from a buffer area in central memory beginning at the address contained in location BA and ending at the address contained in location EA. See Example 2.

Write Record on Disk

```
WRD    N, S, BA, EA, NAME, P
```

Writes from the buffer area in central memory the logical record specified by P of the file identified by NAME onto logical disk N. The words are written, without code translation, from the buffer area beginning at the address contained in location BA and ending at the address contained in EA.

Release Peripheral Processor

```
DRPP   N, S
```

Returns the PP, logical unit N, to the system for general purpose use. This macro must be the final instruction executed before the program completes.

Load Segment

```
LOAD   SYMBOL
```

Loads the subroutine SYMBOL into PP memory. SYMBOL is a subroutine defined by the pseudo opcode SUBP.

6.2 EXAMPLES

EXAMPLE 1

DISPLAY AND TYPEWRITER INPUT/OUTPUT

Suppose a program needs to display a request for control information which requires a reply from the operator. The message might be:

REQUEST SWITCH SETTING 1-5

Either the DHL or DHR macro may be used. Both require that (1) the message data be organized and ready for display before the macro itself is executed, and (2) a set of parameters define the message organization to the operating system.

(1) Data Organization:

Status – A word may be reserved in CM for the status response from the system by use of the BSSCM pseudo code.

```
S     BSSCM    1
```

Data – The message data may be entered into the ASPER program by use of the DPC pseudo code.

```
DATA     DPC

*REQUEST SWITCH
 SETTING 1-5*
```

Before execution of the macro, however, the data must be written into central memory by the CWM instruction. The output data block may be reserved in CM by

```
DIS     BSSCM     3
```

The one-word message input area may be reserved by

```
DISIN     BSSCM     1
```

Record Length – The length of the record to be displayed is 26 characters.

(2) Parameters:

Unit Number – The logical unit number is used only to indicate, relatively, a different console between different macros in the same program. For instance, logical unit number 2 may be any console that is available except one which has been previously referenced as logical unit number 1, 3, 4, etc.

Status – The central memory address, S, may be designated by use of a literal (S). This specifies the first of two consecutive peripheral memory addresses which contain the central memory address S. (Central memory addresses are 18 bits and require two PP 12-bit words).

BA – The location containing the beginning address of the message in central memory, DIS, may be designated, as was S, with a literal (DIS).

EA – The location containing the ending address in central memory, DIS +3, may be similarly written (DIS+3).

RL – The record length may be given explicitly as 26 or as a symbolic PP address, Z, which contains 26.

```
Z     COND     26
```

TAG – A program may put up more than one request which requires a reply from the operator. Therefore an identifier "TAG" is provided. This tag is then appended to the program account number by the system to provide total uniqueness to all requests from the same and/or different programs. Let us suppose the account number is 3512, and TAG is 1.

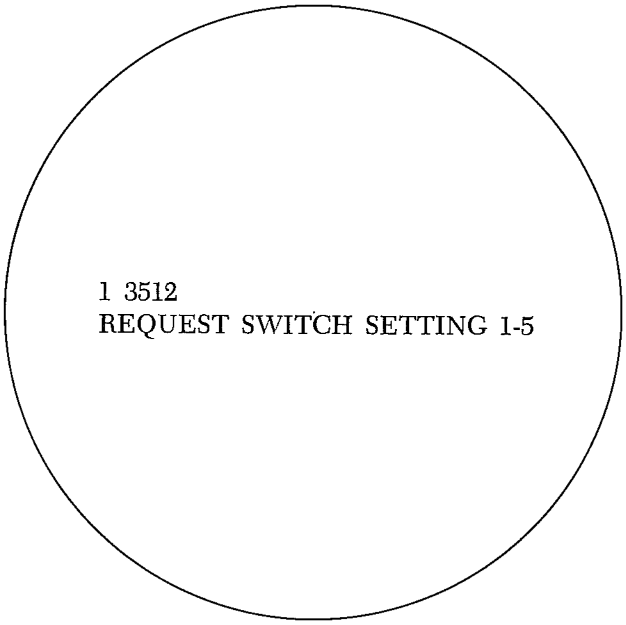
SIZE — The message character size may be chosen as 64, 32, or 16 characters per line. Let us suppose 32 characters per line is chosen.

The macro is then written:

DHL 1, (S), (DIS), (DIS+3), 26, 1, 1

In this example, logical console number 1 is used. The literal notation is used for the address specification of the status response word and data locations, and the RL is given numerically.

The result of executing the macro would be a display positioned somewhere on the left scope of logical console number 1 as follows:



1 3512
REQUEST SWITCH SETTING 1-5

Implication from the message is that the program expects the operator to type a reply. Acceptance of the reply requires another macro, RTY. The parameters for this are N, S, BA, EA, RL, TAG.

N might be 1 to specify the typewriter on logical console 1.

S is a CM word and in this case may be the same one as before.

BA is the beginning address of the input message area, DISIN.

EA need be only one larger than BA since the reply is less than 10 characters.

RL is 1 since the response is a single digit.

TAG is the identifier that the operator must respond to in order to associate his typing with the request being made, namely 1 3512.

The macro issued would be:

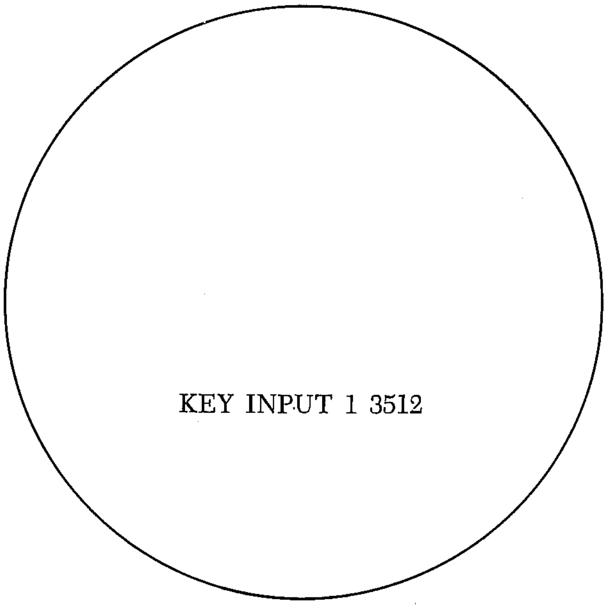
RTY 1, (S), (DISIN), (DISIN+1), 1, 1

When the system indicates a ready with the following display:

the operator must type a number, say 3, which is the switch setting:

3 carriage return

to satisfy both the RTY and DHL macros. The system places this response, 3, at the bottom of the scope.



KEY INPUT 1 3512

EXAMPLE 2

PUNCH, READER, PRINTER, AND TAPE INPUT/OUTPUT

Prior to execution of coded data output macros, it is necessary that the data to be outputed exist in central memory in BCD or display coded form. The coded data are assumed, by the macro, to be packed 10 characters/word from left to right for all words between the addresses contained in locations BA and EA.

Execution of the macro produces r cards, print lines or tape records of 10*(RL) characters each, where r is the number of records required to output all the data between BA and EA. In the process of transfer, each character is translated from the internal code to the output code according to the code conversion mode C.

Suppose data to be punched are:

$1.0_{\Delta\Delta\Delta\Delta}3.925_{\Delta\Delta\Delta}1.3124$ (1)

$2.0_{\Delta\Delta\Delta\Delta}4.177_{\Delta\Delta\Delta}3.2127$

then the internal storage in display code would be:

```
BA     34573300000000365744
       35400000003457363435
       37000000000000000000     (2)
       35573300000000375734
       42420000003657353435
       42000000000000000000
```

The data are to be punched and therefore must be converted to Hollerith which calls for a conversion mode of 1 for display to Hollerith.

The Punch macro is:

```
PCH  1,  (S),  (BA),  (BA+6),  3,  1
```

Execution of the macro would produce the two cards of output left justified from Column 1 as given in (1) above.

For input the same conventions hold except in this case the data are external and will be placed into memory as given above. If the data example above were left justified on two consecutive cards, and the read card macro

```
RDC  1,  (S),  (BA),  (BA+6),  3,  1
```

were executed, the data would come into central memory as shown in (2) above beginning at BA.

Compatibility exists between formats for tape I/O and cards and between card and tape output and printer output. The conversion mode differs due to the introduction of Hollerith code for cards. To print the data in (2) above, the macro used would be:

```
PRN  1,  (S),  (BA),  (BA+6),  3,  2
```

and write tape would be

```
WRC  1,  (S),  (BA),  (BA+6),  3,  2
```

To read the data from the output tape a

```
RFC  1,  (S),  (BA),  (BA+6),  3,  1
```

produces the same internal form as given in (2) above.

For binary data transfers, the conversion mode C = 0 is used. This mode produces a bit-by-bit transfer without conversion to the output device from memory or from the device to memory. In the case of card input and output, one column on the card corresponds to one of 5 12-bit bytes of each CM word. That is, the leftmost 5 columns are inserted from left to right into the first CM word specified, the next five into the next CM word, etc. RL is the number of consecutive 5 column fields to be considered on each card. Examples of binary and coded card inputs and their conversion to card image in central memory are given in Figures 3 and 4,

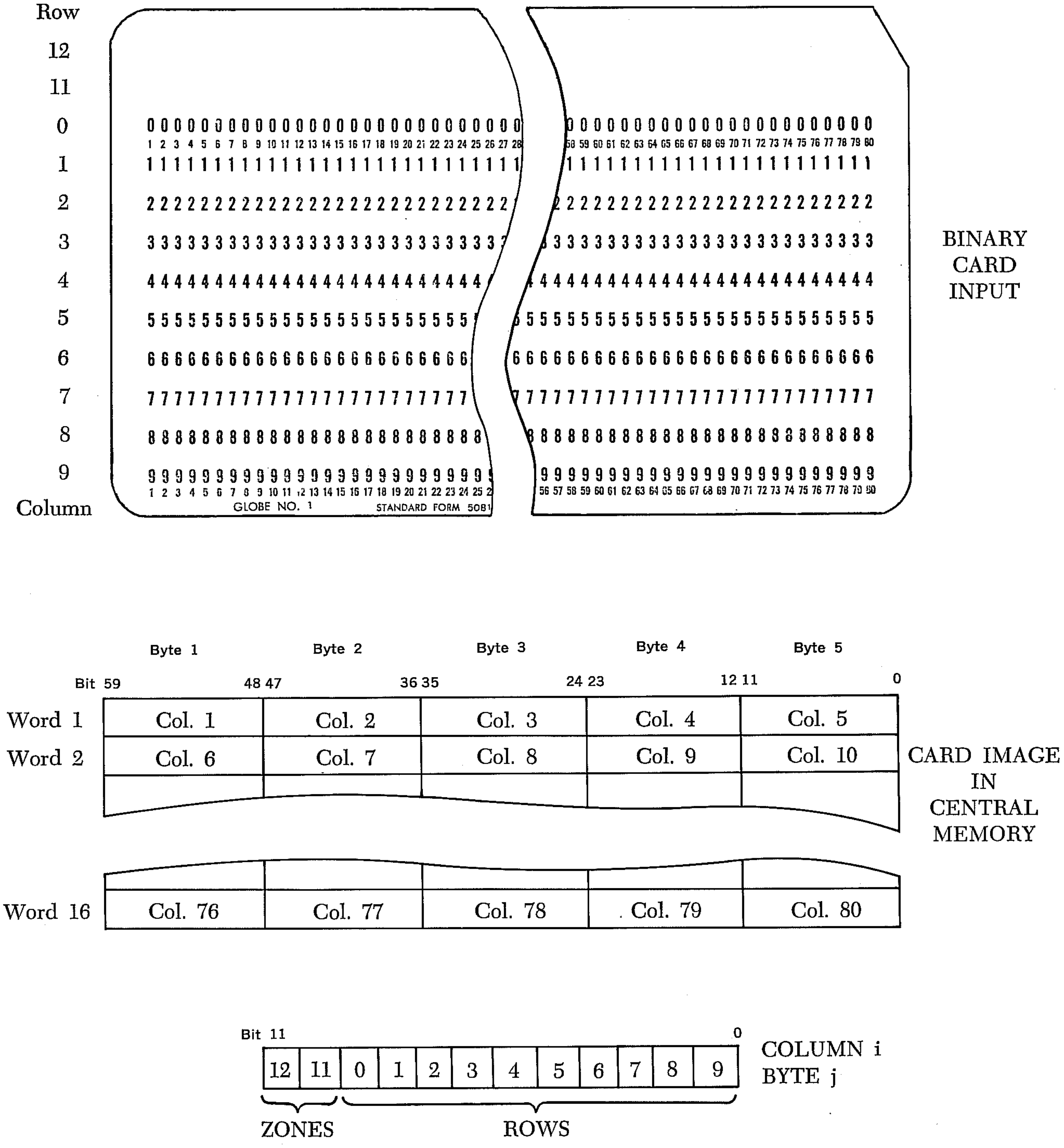
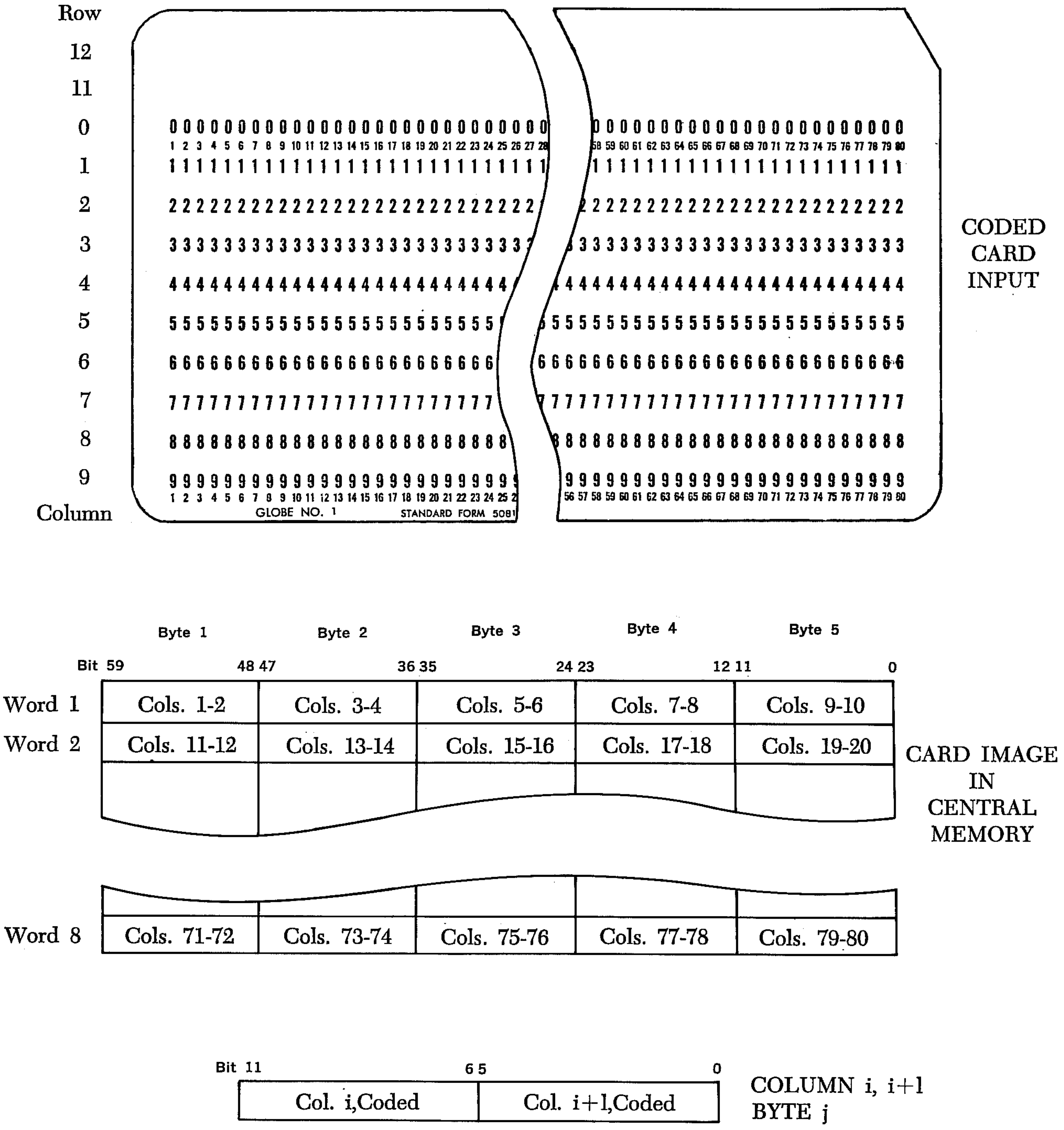


Figure 3 BINARY CARD INPUT



	Byte 1 Bit 59 … 48	Byte 2 47 … 36	Byte 3 35 … 24	Byte 4 23 … 12	Byte 5 11 … 0
Word 1	Cols. 1-2	Cols. 3-4	Cols. 5-6	Cols. 7-8	Cols. 9-10
Word 2	Cols. 11-12	Cols. 13-14	Cols. 15-16	Cols. 17-18	Cols. 19-20
Word 8	Cols. 71-72	Cols. 73-74	Cols. 75-76	Cols. 77-78	Cols. 79-80

Bit 11 … 6	5 … 0
Col. i,Coded	Col. i+1,Coded

Figure 4 CODED CARD INPUT

7. DIAGNOSTICS AND ASSEMBLER OUTPUT

7.1 ASPER ERROR PRINTOUTS

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

C Peripheral Processor Core Overflow. During assembly of this ASPER program, the current location counter exceeds the limit allocated to the object program.

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.

M Multiple 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 is not numeric or included in the set of legal mnemonic codes, pseudos or macros. ASPER forces two words of PSN instructions on this error.

P Parameter List Error. The parameter list does not satisfy ASPER specifications. The list may contain too few or too many parameters.

R Relative Error. The address portion of this relative instruction evaluates out-of-range for this location.

T Tag Error. During the definition of a symbol, the symbol has been found to violate ASPER symbol specifications.

U Undefined Symbol. A reference is made to a symbol that does not appear in the location field. ASPER 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.

7.2 SAMPLE PROGRAM PRINTOUT

```
       0000                   ASPER   PROG1                                   .ROUTINE TO LIST TAPE REVERSE ORDER
       0000                   ORG
0010   0000           IND1    0                                               .INDEX LOCATION 1
0011   0017           IND2    CON     17B                                     .INDEX LOCATION 2
0012   0000           TEMP1   0                                               .TEMP NUMBER 1
0013   0000           TEMP2   0                                               .TEMP NUMBER 2
0014   0000   0001    RECNO   COND    1                                       .RECORD NUMBER FOR DISK MACRO
       0020           INA     BSS     16
       0000                   ORGR
0100   1401           INIT    LDN     1
0101   3415                   STD     RECNO+1                                 .PRESET DISK RECORD NUMBER
0102   0200   7774            RWLW    1,(STAT)                                .REWIND TAPE
0113   0200   7774    START   RFB     1,(STAT),(A),(A+16),16                  .READ TAPE RECORD
0134   0200   7774            WAIW    (STAT),ABORT
0145   2000   0040            LDC     STAT
0147   6016                   CRD     INA                                     .TEST EOF
0150   3016                   LDD     INA
0151   1020                   SHN     16
0152   0603                   PJN     NXT
0153   0100   0221            LJM     EOF                                     .GO TO END-OF-FILE ROUTINE
0155   0200   0325    NXT     RJM     CONV+1                                  .PROCESS RECORD
0157   0200   7774            WRD     1,(STAT),(A),(A+32),FILE1,RECNO         .WRITE DISK
0200   0200   7774            WAIW    (STAT),ABORT                            .CHECK DISK
0211   3615                   AOD     RECNO+1                                 .INCREMENT RECORD NUMBER
0212   1005                   SHN     5
0213   0703                   MJN     *+3
0214   0100   0113            LJM     START                                   .PROCESS IS LESS THAN 4000 RECORDS
0216   2000   0765            LDC     501
0220   3415                   STD     RECNO+1                                 .TOO MANY RECS, TAKE 500
0221   3715           EOF     SOD     RECNO+1
0222   0503                   NJN     *+3
0223   0100   0313            LJM     END
```

```
0225   0200  7774            RDR    1,(STAT),(A),(A+32),FILE1,RECNO    . READ DISK
0246   0200  7774            WAIW   (STAT),ABORT                       . REVERSE ORDER
0257   0200  7774            PRN    1,(STAT),(A),(A+32),8,2            . PRINT 4 LINES
0300   0200  7774            WAIW   (STAT),ABORT
0311   0100  0221            LJM    EOF
0313   0200  7774  END       DRPP   1,(STAT)                           . RELEASE PP AND TERMINATE
0324   0100  0000  CONV      LJM    0                                  . PROCESS RECORD
0326   1420                  LDN    16 $ STD IND1 $ LDN 0 $ STD IND2
0332   3010        CONV1     LDD    IND1 $ MJN CONV                    . PROCESSING DONE
                             .        CODE TO PROCESS RECORD
                             .          OMITTED
                             .        CODE TO PROCESS RECORD
                             .          OMITTED
                             .        CODE TO PROCESS RECORD
                             .          OMITTED
0334   2000  0000            LDC    A
0336   3113                  ADD    TEMP2
0337   6312  0017            CWM    INA+1 TEMP1
0341   3711                  SOD    IND2 $LDN 3 $ RAD TEMP2
0344   0100  0332            LJM    CONV1
       0313        ABORT     EQU    END
       0040        A         BSSCM  32                                 . DEFINE CM AREA
       0001        STAT      BSSCM  1                                  . DEFINE STATUS WORD
       0001                  BSSD   1,32,FILE1,4000                    . DEFINE DISK FILE
                             END
```

7.3 SUMMARY PAGE DIAGNOSTICS

At the end of each ASPER assembly a summary page is printed that includes the number of errors detected, number of symbols assigned, length of ASCENT program, length of ASPER program, amount of central memory storage defined by the ASPER program, and a list of symbols that are undefined, duplicated or not referenced. An example follows:

```
ERRORS     00005
SYMBOLS    00234
ASCENT     02011
ASPER-PP   03121
ASPER-CM   01000
000100  N  ABCDE     000205  N  TAGA     000500  D  AB     002006  U  ST
002007  U      TA     002010  U  SYMB
```

Explanation

ERRORS - Total number of lines with at least one error.

SYMBOLS - The number of symbols assigned a location.

ASCENT - Address of the next central memory location that is available after the central memory program.

ASPER-PP - Address of the next peripheral processor memory location that is available.

ASPER-CM - Number of central memory locations defined by the ASPER program.

aaaaaa NUD TAG

a = location assigned to TAG

N = TAG is not referenced by the program. (NULL)

U = TAG is undefined.

D = TAG is a duplicate symbol.

All numbers are octal.

8. PROGRAM SEGMENTATION

An ASPER program is assumed to be made up of a sequence of one or more segments. The basic segment is defined by the ASPER pseudo operation and any subsegments, if they exist, are defined by SUBP cards (SUBP pseudo operation). The basic segment, along with the other subsegments, exists in central memory as binary card images, having been placed there by the job loader. The loading of the basic segment into a PP is initiated by the CM portion of the program through the use of the TPP macro. As soon as the basic segment is loaded, execution of that segment begins in the PP. It is the responsibility of the ASPER program (basic segment) to provide overlay requests where needed. The overlay is initiated by issuance of the LOAD macro by the ASPER program. The location at which the subsegment is to be loaded was defined at assembly time by the SUBP pseudo operation.

The basic segment of a program is defined with an ASPER header card:

```
ASPER     P1
```

where: P1 is the name of a program.

Coding within the segment follows normal rules as described in Sections 1 through 5. When all of the coding for the basic segment is completed, the subprogram pseudo operation (SUBP) is used to define any additional subsegment to be overlayed.

```
SUBP     P1, P2
```

where: P1 is the symbolic name of the new segment to be used by the overlay request.
P2 is an operand which defines the overlay point within the first segment and is of the form:

SYMBOL

SYMBOL ± CONSTANT

CONSTANT

Once the basic segment is in execution, any other subsegment may be called by using the LOAD pseudo operation.

```
LOAD     P1
```

where: P1 is the name of the subsegment wanted.

No limit is placed on the number of subsegments allowed in a routine. The limit on the size of usable peripheral processor core is 62 direct locations, $2\text{-}77_8$ and 3520 non-direct locations, $100_8\text{-}6777_8$, with the other locations being reserved for PP resident routines.

```
        LOC      OPCODE          ADDRESS                   REMARKS
Col. 2        9  11

                 ASPER           NAME                      . Define routine name.
                 .                                         . Normal coding and decision
                 .                                         . to load SEG1.
                 .
                 LOAD            SEG1                      . Loads SEG1 into PP.
                 .
                 RJM             SAM+1                     . Stores P+2 in SAM+1 for
                                                           . return from SEG1.
                  .
                 .
                 .                                         . Normal coding and decision
                 .                                         . to load SEG3.
                 LOAD            SEG3                      . Loads SEG3 into PP.
                 .
                 LJM             TOM                       . Non-return entry into SEG3.
                 .
                 .
        TAG1     .                                         . Overlay point for SEG1.
                 .
        TAG3     .    .                                    . Overlay point for SEG3.
                 .    .
                      .
                      .
        TAG2     .  . .                                    . Overlay point for SEG2.
                 .  . .
                 .  .
                 .  .
                 SUBP            SEG1, TAG1                . Defines segment name and load point.
        SAM      LJM             0                         . Address is filled by RJM instruction.
                 .
                 .                                         . Segment coding
                 LOAD            SEG2                      . Loads SEG2 into PP.

                 LJM             SAM                       . Return to main program.
                 SUBP            SEG3, TAG3                . Defines segment name and load point.
        TOM           .
                      .                                    . Segment coding
                 DRPP            N, STAT                   . Release PP to system.
                 SUBP            SEG2, TAG2                . Defines segment name and load point.
                   .
                   .
                   .
                 END                                       . End of program
```

NOTES:

1. In this illustration, SEG 2 is a non-executable segment, containing input data only.
2. Both RJM and LJM use two computer words since they require a 12-bit address and a 6-bit index designator, the address being placed in the second word.
3. The pseudo opcodes, ASPER and SUBP, are non-executable instructions and cause no binary word to be generated. However, they do cause the assembler to generate control information in the form of control cards in the binary object program.

APPENDIX

TABLE 1

PERIPHERAL PROCESSOR OPERATION CODES

OCTAL OPCODE	MNEMONIC	ADDRESS	COMMENTS
00	PSN		. Pass
01	LJM	m d	. Long jump to m + (d)
02	RJM	m d	. Return jump to m + (d)
03	UJN	d	. Unconditional jump d
04	ZJN	d	. Zero jump d
05	NJN	d	. Nonzero jump d
06	PJN	d	. Plus jump d
07	MJN	d	. Minus jump d
10	SHN	d	. Shift d
11	LMN	d	. Logical difference d
12	LPN	d	. Logical product d
13	SCN	d	. Selective clear d
14	LDN	d	. Load d
15	LCN	d	. Load complement d
16	ADN	d	. Add d
17	SBN	d	. Subtract d
20	LDC	dm	. Load dm
21	ADC	dm	. Add dm
22	LPC	dm	. Logical product dm
23	LMC	dm	. Logical difference dm
24	PSN		. Pass
25	PSN		. Pass
26	EXN		. Exchange jump
27	RPN		. Read program address
30	LDD	d	. Load (d)
31	ADD	d	. Add (d)
32	SBD	d	. Subtract (d)
33	LMD	d	. Logical difference (d)
34	STD	d	. Store (d)
35	RAD	d	. Replace add (d)
36	AOD	d	. Replace add one (d)
37	SOD	d	. Replace subtract one (d)
40	LDI	d	. Load ((d))
41	ADI	d	. Add ((d))
42	SBI	d	. Subtract ((d))
43	LMI	d	. Logical difference ((d))
44	STI	d	. Store ((d))
45	RAI	d	. Replace add ((d))
46	AOI	d	. Replace add one ((d))
47	SOI	d	. Replace subtract one ((d))
50	LDM	m d	. Load (m + (d))
51	ADM	m d	. Add (m + (d))

OCTAL OPCODE	MNEMONIC	ADDRESS	COMMENTS
52	SBM	m d	. Subtract m + (d))
53	LMM	m d	. Logical difference (m + (d))
54	STM	m d	. Store (m + (d))
55	RAM	m d	. Replace add (m + (d))
56	AOM	m d	. Replace add one (m + (d))
57	SOM	m d	. Replace subtract one (m + (d))
60	CRD	d	. Central read from (A) to d
61	CRM	m d	. Central read (d) words from (A) to m
62	CWD	d	. Central write to (A) from d
63	CWM	m d	. Central write (d) words to (A) from m
64	AJM	m d	. Jump to m if channel d active
65	IJM	m d	. Jump to m if channel d inactive
66	FJM	m d	. Jump to m if channel d full
67	EJM	m d	. Jump to m if channel d empty
70	IAN	d	. Input to A from channel d
71	IAM	m d	. Input (A) words to m from channel d
72	OAN	d	. Output from A on channel d
73	OAM	m d	. Output (A) words from m on channel d
74	ACN	d	. Activate channel d
75	DCN	d	. Disconnect channel d
76	FAN	d	. Function (A) on channel d
77	FNC	m d	. Function m on channel d

NOTES TO TABLE 1

NOTATION	INTERPRETATION
d	Implies d itself
(d)	Implies the contents of d
((d))	Implies the contents of the location specified by d
m	Implies m itself used as an address
m + (d)	The contents of d are added to m to form an operand (jump address)
(m + (d))	The contents of d are added to m to form the address of the operand
dm	Implies an 18-bit quantity with d as the upper 6 bits and m as the lower 12 bits

TABLE 2

PSEUDO OPERATION CODES

OPCODE	MEANING
ASPER	Defines PP program
SUBP	Defines overlay
ORG	Assigns program words to direct locations, nonrelocatable
ORGR	Assigns program words to nondirect locations, relocatable
BSSD	Reserves disk space
BSS	Reserves peripheral memory region
BSSZ	Reserves peripheral memory region and presets it to zero
BSSCM	Reserves central memory region
EQU	Equates a symbol to a value
DPC	Inserts display-coded characters into program
BCD	Inserts BCD characters into program
CON	Constructs 12-bit constants
COND	Constructs 18-bit constants
END	Defines end of PP program
LIST	Controls side-by-side listing
SPACE	Spaces side-by-side listing
EJECT	Ejects page on side-by-side listing

TABLE 3

SYSTEM MACROS

RQTW	Request tape assignment from system.
DRTW	Release tape back to system.
SFFW	Search file mark forward.
SFBW	Search file mark backward.
WFMW	Write file mark.
RWLW	Rewind tape to load point.
RWUW	Rewind tape for unload.
FSPW	Forespace.
BSPW	Backspace.
RFCW	Read tape forward coded mode.
RFBW	Read tape forward binary mode.
WRCW	Write tape coded mode.
WRBW	Write tape binary mode.
RDHW	Read record and hold data on disk.
RDRW	Read record and release data on disk.
WRDW	Write record on disk.
SSPW	Single space printer.
DSPW	Double space printer.
FC7W	Select Format Channel 7.
FC8W	Select Format Channel 8.
MC1W	Select Monitor Channel 1.
MC2W	Select Monitor Channel 2.
MC3W	Select Monitor Channel 3.
MC4W	Select Monitor Channel 4.
MC5W	Select Monitor Channel 5.
MC6W	Select Monitor Channel 6.
CMCW	Clear Monitor Channels 1 – 6.
SPAW	Suppress space after next print.
PRNW	Print single line or multiple lines.
PCHW	Punch cards.
RDCW	Read cards.
DSRW	Display on right scope for system time limit.
DSLW	Display on left scope for system time limit.
DHRW	Display on right scope and hold indefinitely.
DHLW	Display on left scope and hold indefinitely.
RDPW	Remove display.
RTYW	Read console typewriter.
WAIW	Check status word.
TPPW	Transfer program SYMBOL from CM to PP memory and begin execution with first ASPER instruction.
RQMW	Request memory.
DRMW	Release memory.
RQDW	Request disk space.
DRDW	Release disk space.
RQCW	Request I/O channel.
DRCW	Release I/O channel.
DRPP	Release peripheral processor.
LOAD	Load segment SYMBOL.

TABLE 4
6600 COMPUTER CHARACTER CODES

Character	*Display Code*	*Printer Code*	*Hollerith Punch Positions*
A	01	61	12-1
B	02	62	12-2
C	03	63	12-3
D	04	64	12-4
E	05	65	12-5
F	06	66	12-6
G	07	67	12-7
H	10	70	12-8
I	11	71	12-9
J	12	41	11-1
K	13	42	11-2
L	14	43	11-3
M	15	44	11-4
N	16	45	11-5
O	17	46	11-6
P	20	47	11-7
Q	21	50	11-8
R	22	51	11-9
S	23	22	0-2
T	24	22	0-3
U	25	24	0.4
V	26	25	0-5
W	27	26	0.6
X	30	27	0.7
Y	31	30	0-8
Z	32	31	0-9
0	33	12	0
1	34	01	1
2	35	02	2
3	36	03	3
4	37	04	4
5	40	05	5
6	41	06	6
7	42	07	7
8	43	10	8
9	44	11	9
blank	00	20	space
+	45	60	12
−	46	40	11
*	47	54	11-8-4
/	50	21	0-1
(	51	34	0-8-4

Character	*Display Code*	*Printer Code*	*Hollerith Punch Positions*
)	52	74	12-8-4
=	54	13	8-3
≠	55	14	8-4
,	56	33	0-8-3
.	57	73	12-8-3
$	63	34	11-8-3
:		00	
≤		15	
%		16	
[		17	
]		32	
→		35	
≡		36	
∧		37	
∨		52	
↑		55	
↓		56	
>		57	
<		72	
≥		75	
¬		76	
;		77	

Central Processor
Instruction Execution Times

Mnemonic & Octal Code		Name	Time (Minor Cycles)
		BRANCH UNIT	
PS	00	STOP	–
RJ	01	RETURN JUMP to K	13
JP	02	GO TO K + Bi	8*
ZR	030	GO TO K if Xj = zero	8*
NZ	031	GO TO K if Xj ≠ zero	8*
PL	032	GO TO K if Xj = positive	8*
NG	033	GO TO K if Xj = negative	8*
IR	034	GO TO K if Xj is in range	8*
OR	035	GO TO K if Xj is out of range	8*
DF	036	GO TO K if Xj is definite	8*
ID	037	GO TO K if Xj is indefinite	8*
EQ ZR	04 04	GO TO K if Bi = Bj	8*
NE NZ	05 05	GO TO K if Bi ≠ Bj	8*
GE PL	06 06	GO TO K if Bi ≧ Bj	8*
LT NG	07 07	GO TO K if Bi < Bj	8*
		BOOLEAN UNIT	
BXi	10	TRANSMIT Xj to Xi	3
BXi	11	LOGICAL PRODUCT of Xj and Xk to Xi	3
BXi	12	LOGICAL SUM of Xj and Xk to Xi	3
BXi	13	LOGICAL DIFFERENCE of Xj and Xk to Xi	3
BXi	14	TRANSMIT Xk COMP. to Xi	3
BXi	15	LOGICAL PRODUCT of Xj and Xk COMP. to XI	3
BXi	16	LOGICAL SUM of Xj and Xk COMP. to Xi	3
BXi	17	LOGICAL DIFFERENCE of Xj and Xk COMP. to Xi	3
		SHIFT UNIT	
LXi	20	SHIFT Xi LEFT jk places	3
AXi	21	SHIFT Xi RIGHT jk places	3
LXi	22	SHIFT Xk NOMINALLY LEFT Bj places to Xi RXi	3
AXi	23	SHIFT Xk NOMINALLY RIGHT Bj places to Xi	3
NXi	24	NORMALIZE Xk in Xi and Bj	4
ZXi	25	ROUND AND NORMALIZE Xk in Xi and Bj	4
UXi	26	UNPACK Xk to Xi and Bj	3
PXi	27	PACK Xi from Xk and Bj	3
MXi	43	FORM jk MASK in Xi	3
		ADD UNIT	
FXi	30	FLOATING SUM of Xj and Xk to Xi	4
FXi	31	FLOATING DIFFERENCE of Xj and Xk to Xi	4
DXi	32	FLOATING DP SUM of Xj and Xk to Xi	4
DXi	33	FLOATING DP DIFFERENCE of Xj and Xk to Xi	4
RXi	34	ROUND FLOATING SUM of Xj and Xk to Xi	4
RXi	35	ROUND FLOATING DIFFERENCE of Xj and Xk to Xi	4

Mnemonic & Octal Code		Name	Time (Minor Cycles)
		LONG ADD UNIT	
IXi	36	INTEGER SUM of Xj and Xk to Xi	3
IXi	37	INTEGER DIFFERENCE of Xj and Xk to Xi	3
		MULTIPLY UNIT	
FXi	40	FLOATING PRODUCT of Xj and Xk to Xi	10
RXi	41	ROUND FLOATING PRODUCT of Xj and Xk to Xi	10
DXi	42	FLOATING DP PRODUCT of Xj and Xk to Xi	10
		DIVIDE UNIT	
FXi	44	FLOATING DIVIDE Xj by Xk to Xi	29
RXi	45	ROUND FLOATING DIVIDE Xj by Xk to Xi	29
NO	46	PASS	–
CXi	47	SUM of 1's in Xk to Xi	8
		INCREMENT UNIT	
SAi	50	SUM of Aj and K to Ai	3
SAi	51	SUM of Bj and K to Ai	3
SAi	52	SUM of Xj and K to Ai	3
SAi	53	SUM of Xj and Bk to Ai	3
SAi	54	SUM of Aj and Bk to Ai	3
SAi	55	DIFFERENCE of Aj and Bk to Ai	3
SAi	56	SUM of Bj and Bk to Ai	3
SAi	57	DIFFERENCE of Bj and Bk to Ai	3
SBi	60	SUM of Aj and K to Bi	3
SBi	61	SUM of Bj and K to Bi	3
SBi	62	SUM of Xj and K to Bi	3
SBi	63	SUM of Xj and Bk to Bi	3
SBi	64	SUM of Aj and Bk to Bi	3
SBi	65	DIFFERENCE of Aj and Bk to Bi	3
SBi	66	SUM of Bj and Bk to Bi	3
SBi	67	DIFFERENCE of Bj and Bk to Bi	3
SXi	70	SUM of Aj and K to Xi	3
SXi	71	SUM of Bj and K to Xi	3
SXi	72	SUM of Xj and K to Xi	3
SXi	73	SUM of Xj and Bk to Xi	3
SXi	74	SUM of Aj and Bk to Xi	3
SXi	75	DIFFERENCE of Aj and Bk to Xi	3
SXi	76	SUM of Bj and Bk to Xi	3
SXi	77	DIFFERENCE of Bj and Bk to Xi	3

Comp.—Complement

DP—Double Precision

*Add 5 minor cycles to branch time for a branch to an instruction which is out of the stack (no memory conflict considered)

CONTROL DATA SALES OFFICES

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

ATHENS • CANBERRA • DUSSELDORF • FRANKFURT • THE HAGUE • HAMBURG JOHANNESBURG • LONDON • MELBOURNE • MEXICO CITY (REGAL ELECTRONICA DE MEXICO, S.A.) • MILAN • MONTREAL • MUNICH • OSLO • OTTAWA PARIS • SAVYON • STOCKHOLM • STUTTGART • SYDNEY • TOKYO (C. ITOH ELECTRONIC COMPUTING SERVICE CO., LTD.) • TORONTO • ZURICH

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

Pub. No. 60101700B

Litho in U.S.A.