TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

OP CODE DIRECTORY
BY CODE

OP	Mnemonic	Section	Compatibility Notes
01	INC	8.1	A.08
02	ADD	8.2	A.09
03	DEC	8.3	A.08
04	SUB	8.4	A.09
05	MPY	8.5	A.10
06	DIV	8.6	
08	MVD	14.1	A.20
09	MVL	14.2	A.21
10	MVA	14.3	A.03
11	MVN	14.4	A.04
12	MVW	14.5	A.07
13	MVC	14.6	A.07
14	MVR	14.7	A.15
15	TRN	14.8	A.14
16	SDE	15.1	
17	SDU	15.2	
18	SZE	15.3	
19	SZU	15.4	
20	NOP	11.1	
21	LSS	11.1	A.06
22	EQL	11.1	A.06
23	LEQ	11.1	A.06
24	GTR	11.1	A.06
25	NEQ	11.1	A.06
26	GEQ	11.1	A.06
27	BUN	11.1	
28	OFL	11.1	
29	HBR	12.1	
2A	NUL	11.1	A.06
2B	GTN	11.1	A.06
30	BCT	13.1	
31	NTR	13.2	A.19
32	EXT	13.3	
33	BRT	15.14	A.22
34	BST	15.15	A.22
35	VEN	13.4	
37	SLL	15.6	A.26
38	SLD	15.7	A.26
39	SEA	15.5	A.12
40	BZT	15.10	A.05

OP CODE DIRECTORY
BY CODE

OP	Mnemonic	Section	Compatibility Notes
41	BOT	15.11	A.05
42	AND	15.16	A.11
43	ORR	15.17	A.11
44	NOT	15.18	A.11
45	CPA	15.12	A.02
46	CPN	15.13	
47	SMF	20.8	
48	HBK	12.2	A.25
49	EDT	14.9	A.13
50	IAD	9.3	
51	IAS	9.4	
52	ISU	9.5	
53	ISS	9.6	
54	IMU	9.7	
55	IMS	9.8	
57	IMI	9.9	
58	ILD	9.1	
59	IST	9.2	
60	LOK	20.4	
61	ASP	13.7	
62	HCL	13.5	
63	RET	13.6	
64	SLT	15.8	
65	WHR	20.7	
66	STB	15.9	
67	LIX	20.2	
68	SIX	20.3	
69	ILS	20.5	
6A	MLS	20.6	
70	RAA	10.3	A.17, 18
71	RAS	10.4	A.17, 18
72	RSU	10.5	A.17, 18
73	RSS	10.6	A.17, 18
74	RMU	10.7	A.17
75	RMS	10.8	A.17
76	RDV	10.9	
77	RDS	10.10	
78	RLD	10.1	
79	RST	10.2	
84	ACM	10.11	

OP CODE DIRECTORY
BY CODE

OP	Mnemonic	Section	Compatibility Notes
85	CIO	16.4	
86	ATE	20.1	
87	MOP	19.1	
88	D2B	17.1	
89	B2D	17.2	
90	INT	13.8	
91	SRD	16.3	A.23
92	RAD	16.2	
93	BRV	13.9	
94	IIO	16.1	
95	RDT	18.1	
97	STT	18.2	
98	IOC	16.5	
99	SST	20.10	A.40
A0	MVS	21.1	
A1	CPS	21.2	
A2	HSH	21.3	
AB	BAD	20.9	

OP CODE DIRECTORY
BY MNEMONIC

Mnemonic	OP	Section	Compatibility Notes
ACM	84	10.11	
ADD	02	8.2	A.09
AND	42	16.14	A.10
ASP	61	13.7	
ATE	86	20.1	
BAD	AB	20.9	
BCT	30	13.1	
BOT	41	15.11	A.05
BRT	33	15.14	A.22
BRV	93	13.9	
BST	34	15.15	A.22
BUN	27	11.1	
BZT	40	15.10	A.05
B2D	89	17.2	
CIO	85	16.4	
CPA	45	15.12	A.02
CPN	46	15.13	
CPS	A1	21.2	
DEC	03	8.3	A.08
DIV	06	8.6	
D2B	88	17.1	
EDT	49	14.9	A.13
EQL	22	11.1	A.06
EXT	32	13.3	
GEQ	26	11.1	A.06
GTN	2B	11.1	A.06
GTR	24	11.1	A.06
HBK	48	12.2	A.25
HBR	29	12.1	
HCL	62	13.5	
HSH	A2	21.3	
IAD	50	9.3	
IAS	51	9.4	
IIO	94	16.1	
ILD	58	9.1	
ILS	69	20.5	
IMI	57	9.9	
IMS	55	9.8	
IMU	54	9.7	
INC	01	8.1	A.08

OP CODE DIRECTORY
BY MNEMONIC

Mnemonic	OP	Section	Compatibility Notes
INT	90	13.8	
IOC	98	16.5	
ISS	53	9.6	
IST	59	9.2	
ISU	52	9.5	
LEQ	23	11.1	A.06
LIX	67	20.2	
LOK	60	20.4	
LSS	21	11.1	A.06
MLS	6A	20.6	
MOP	87	19.1	
MPY	05	8.5	A.10
MVA	10	14.3	A.03
MVC	13	14.6	A.07
MVD	08	14.1	A.20
MVL	09	14.2	A.21
MVN	11	14.4	A.04
MVR	14	15.7	A.15
MVS	A0	21.1	
MVW	12	14.5	A.07
NEQ	25	11.1	A.06
NOP	20	11.1	
NOT	44	15.18	A.11
NTR	31	13.2	A.19
NUL	2A	11.1	A.06
OFL	28	11.1	
ORR	43	15.17	A.11
RAA	70	10.3	A.17, 18
RAD	92	16.2	
RAS	71	10.4	A.17, 18
RDS	77	10.10	
RDT	95	18.1	
RDV	76	10.9	
RET	63	13.6	
RLD	78	10.1	
RMS	75	10.8	A.17
RMU	74	10.7	A.17
RSS	73	10.6	A.17, 18
RST	79	10.2	
RSU	72	10.5	A.17, 18

OP CODE DIRECTORY
BY MNEMONIC

Mnemonic	OP	Section	Compatibility Notes
SDE	16	15.1	
SDU	17	15.2	
SEA	39	15.5	A.12
SIX	68	20.3	
SLD	38	15.7	A.26
SLL	37	15.6	A.26
SLT	64	15.8	
SMF	47	20.8	
SRD	91	16.3	A.23
SST	99	20.10	A.40
STB	66	15.9	
STT	97	18.2	
SUB	04	8.4	A.09
SZE	18	15.3	
SZU	19	15.4	
TRN	15	14.8	A.14
VEN	35	13.4	
WHR	65	20.7	

1 INTRODUCTION

1.1 PREFACE

This specification defines the instruction set for the V-Series processor family.

Appendix A - Compatibility Notes: Describes the machine dependent variations to the instruction specifications.

1.2 RELATED SPECIFICATIONS

See Appendix A - Compatibility Notes (A.01).

1.3 GENERAL DESCRIPTION

This instruction set consists of powerful, high level variable length instructions of up to three operand addresses each. It has instructions for data manipulation and local program control as well as more complex instructions to address outside of the local program environment and switch program environments. This family of instructions will also permit the system to address memory of greater than five million bytes. Addressing limitations are machine dependent. See Appendix A - Compatibility Notes (A.53).

Operands may be fixed in length, may vary from 1 to 100 units, or may have their lengths defined by a begin/end address pair. In addition, data types may specify unsigned numeric, signed numeric, or unsigned alpha.

2 DATA REPRESENTATION

Conversion between signed and unsigned 4-bit representation and 8-bit representation is accomplished automatically during the execution of instructions.

2.1 SIGNED NUMERIC FORMAT (SN)

Data is interpreted in units of 4 bits (one digit). The sign is interpreted as a separate and leading 4-bit unit. The 4-bit code is interpreted by the arithmetic units as follows:

4-bit Code	Digit	Sign Code
0000	0	+
0001	1	+
0010	2	+
0011	3	+
0100	4	+
0101	5	+
0110	6	+
0111	7	+
1000	8	+
1001	9	+
1010	A Undefined*	+
1011	B Undefined*	+
1100	C Undefined*	+
1101	D Undefined*	-
1110	E Undefined*	+
1111	F Undefined*	+

* Undefined - The hexadecimal digits A through F are hereafter referred to as "undigits". Use of undigits in an arithmetic operand, except for the sign digit, will cause an Invalid Arithmetic Data Fault. See Appendix A - Compatibility Notes (A.16).

2.1 SIGNED NUMERIC FORMAT (SN) (Continued)

When the result of an operation is signed, the sign-digit is stored as follows:

+ = 1100 (C)
- = 1101 (D)

When the sign digit is interpreted as PLUS, it will compare as HIGH relative to a sign digit interpreted as MINUS.

In the examples that are given at the end of the description of many of the instructions the plus symbol (+) is used to indicate a sign digit value of 0-9, A, B, C, E or F. The symbol also is used to represent a plus sign value that may be stored within the processor. The letter "C" is used to indicate that the processor has written a plus sign into memory. The letter "D" is used to indicate the minus sign except that the minus symbol (-) is used to represent a minus sign value that may be stored within the processor.

2.2 UNSIGNED NUMERIC FORMAT (UN)

Data is interpreted in units of four-bits (one digit). Unsigned data fields are assumed to be positive.

2.3 UNSIGNED ALPHA FORMAT (UA)

Data is interpreted in units of eight bits (one byte or one character).

The internal representation of alpha data is in the Extended Binary Coded Decimal Interchange Code (EBCDIC).

Eight-bit data is considered unsigned except in the case of the Move Alpha (OP = 10), Move Numeric (OP = 11), and Edit (OP = 49) instructions. Additional details are given in the description of these instructions.

Alphanumeric comparisons are binary and thus the (low-to-high) collating sequence for EBCDIC is symbols-alphas-digits.

3 INSTRUCTION REPRESENTATION

"Reserved" or "not specified" bits, digits, or characters must be zero and are reserved for future specification. Ignored bits, digits or characters are not examined and may be any value.

All fields are addressed most significant digit first, unless specifically noted otherwise.

All instructions must start at an even address or cause an Address Error Fault (AEX = 43).

The data fields are called the A-field, the B-field and the C-field. AF and BF generally refer to the lengths of A and B fields respectively The address of each field is called the A, B and C address. The data type of each field is generally defined by the first digit (controller bits) of the A, B and C addresses.

3.1 INSTRUCTION FORMAT

The Processor instructions may vary in length from 4 to 30 digits. An instruction may use a mixture of Extended Address and Non-extended Address formats. Extended format is specified by the value of the second digit of each of the A, B, and C address syllables. An extended address occupies 8 digits whereas a non-extended address occupies 6 digits of an instruction.

Non-Extended Format:

```
Description
===================================
OP VV
OP VVVV
OP AAAA
OP AAAAAA
OP AFBF AAAAAA
OP AFBF AAAAAA BBBBBB
OP AFBF AAAAAA BBBBBB CCCCCC
===================================
```

Extended Format:

```
Description
===================================
OP AAAAAAAA
OP AFBF AAAAAAAA
OP AFBF AAAAAAAA BBBBBBBB
OP AFBF AAAAAAAA BBBBBBBB CCCCCCCC
===================================
```

Where:

OP	= Operator Code
V	= Variant Digits
AFBF	= A and B-field length variant.
A, B, C	= Address of respective data fields

3.1.1 NON-EXTENDED FORMAT

Non-extended direct addressing capability is from 0 through 99,999. In a Branch Instruction, the two address controller bits can be used to extend the address range to 299,998 (See Section 3.5). The non-extended format is shown in Figure 3.1-1.

3.1.2 EXTENDED FORMAT

Extended direct addressing capability is from 0 through 999,999.

An Extended Indicant is specified if the two high order bits of the second most significant digit of an address syllable are true. An Extended Indicant signifies that the next six digits contain the address and determines which index registers may be specified for this address. See Section 3.2. The non-extended and extended addressing formats are shown in Figure 3.1-1.

FIGURE 3.1-1 NON-EXTENDED/EXTENDED ADDRESSING FORMAT

```
                                +-----+----+----+----+----+----+
    NON-EXTENDED ADDRESS        |xx xx| 5  | 4  | 3  | 2  | 1  |
                                +-----+----+----+----+----+----+
                                  |  |   |    |    |    |    |
         INDEX INDICANT ----------   |   |    |    |    |    |
                                     |   |    |    |    |    |
         ADDRESS CONTROLLER ----------   |    |    |    |    |
                                         |    |    |    |    |
         ADDRESS TEN THOUSANDS ----------|    |    |    |    |
                                         |    |    |    |    |
         ADDRESS THOUSANDS --------------|----|    |    |    |
                                         |    |    |    |    |
         ADDRESS HUNDREDS ---------------|----|----|    |    |
                                         |    |    |    |    |
         ADDRESS TENS -------------------|----|----|----|    |
                                         |    |    |    |    |
         ADDRESS UNITS ------------------|----|----|----|----|
                                         |    |    |    |    |
                                         |    |    |    |    |
                       +-----+-----+----+----+----+----+----+----+
    EXTENDED ADDRESS   |xx xx|  X  | 6  | 5  | 4  | 3  | 2  | 1  |
                       +-----+-----+----+----+----+----+----+----+
                         |  |   |    |
    INDEX INDICANT -------  |   |    |
                            |   |    |
    ADDRESS CONTROLLER ------   |    |
                                |    |
    EXTENDED INDICANT ----------|    |
                                     |
    ADDRESS HUNDRED THOUSANDS --------
```

3.2 ADDRESS RESOLUTION

Under the MCP for the V Series machines, most of the MCP and all user programs are partitioned into a number of separate memory pieces. Each is defined by its base and limit which are always MOD 1000. All addresses to this memory are relative to one of these base and limits.

All processes running under the MCP on these machines have accessability to memory via eight base and limit pairs. Base #0 is defined as that process' context (data) area. Base #1 is defined as that process' code area. Non-indexed addresses will refer to base #0 or base #1, depending upon whether the address refers to data or code, respectively. In order to support user programs with only one base and limit (i.e. with the intermixed code and data), both base #0 and base #1 point to the same area of memory.

Processes can also address memory via all 8 base and limit pairs through the use of the base indicant digit in one of the index registers. With non-extended addressing, IX1, IX2, and IX3 can be used. Extended addressing also allows the use of IX4, IX5, IX6, and IX7.

Addresses will be resolved according to the following chart:

EXTENDED INDICANT	INDEX INDICANT	ADDRESSING MODE
None	0	Context Relative
None	1	IX1 w/Base Indicant
None	2	IX2 w/Base Indicant
None	3	IX3 w/Base Indicant
A	0	Address Error
A	1	Address Error
A	2	Address Error
A	3	Address Error
B	0	Address Error
B	1	Address Error
B	2	Address Error
B	3	Address Error

3.2 ADDRESS RESOLUTION (Continued)

EXTENDED INDICANT	INDEX INDICANT	ADDRESSING MODE
C	0	Context Relative
C	1	IX1 w/Base Indicant
C	2	IX2 w/Base Indicant
C	3	IX3 w/Base Indicant
D	0	Code Base Relative (Base #1)
D	1	IX4 w/Base Indicant
D	2	IX5 w/Base Indicant
D	3	IX6 w/Base Indicant
E	0	Address Error
E	1	IX7 w/Base Indicant
E	2	Address Error
E	3	Address Error
F	0	Address Error
F	1	Address Error
F	2	Address Error
F	3	Address Error

3.2.1 NON-INDIRECT ADDRESS

A non-indirect context relative address is relative to the Code Base (Base #1) for the following instruction opcodes:

OP	Mnemonic	Name
20	NOP	No-operation
21	LSS	Branch on Less
22	EQL	Branch on Equal
23	LEQ	Branch on Less or Equal
24	GTR	Branch on Greater
25	NEQ	Branch on Not Equal
26	GEQ	Branch on Gtr or Eql
27	BUN	Branch Unconditional
28	OVF	Branch on Overflow
29	HBR	Halt Branch
2A	NUL	Branch on Null
2B	GTN	Branch on Gtr or Null
31	NTR	Enter
32	EXT	Exit

A non-indirect context relative address is relative to the Data Base (Base #0) for all other instruction opcodes.

Address digits are limited to the decimal values of 0-9. Undigits (A through F) in a resolved final address will cause an Address Error fault.

These specifications define certain absolute address fields, however, some machines require fixed address modifications. See Appendix A - Compatibility Notes (A.35).

3.2.2 INDIRECT ADDRESS

The referenced address field does not contain the operand data, but contains another address. The latter address may point to data or still another address. This indirect reference may be carried to any depth. The controller of the final (direct) address specifies the format of the operand field to be accessed and must conform to any address controller restrictions for the instruction.

Full generality of indexing is maintained in indirect addressing. Any or all of the indirect addresses in a chain may be indexed. An address is always indexed before the indirect reference is taken.

Extended addressing may be applied to any or all of the indirect addresses in a chain.

All Indirect Addresses in an indirect address chain that are context relative are relative to the Data Base (Base #0). If the indirect address is indexed, the specified Base Indicant from the index register will be used to determine the base. If the indirect address is extended with a "D" and no index register is specified, the address is relative to the Code Base (Base #1).

Undigits in an unresolved intermediate address will produce an Address Error Fault (AEX = 32). See Appendix A - Compatibility Notes (A.29).

3.2.3 BRANCH ADDRESS

If non-extended address format is used, the branch address in the Address Branch, Halt Branch, Enter and Exit instructions have a maximum address capability of 299,998. To accomplish this the address controller bits carry the following significance:

00 = 0 = Most Significant Digit of Address

01 = 1 = Most Significant Digit of Address

10 = 2 = Most Significant Digit of Address

11 = 3 = Indirect Address

If address extension is used, the address controller in the Branch, Halt Branch, Enter and Exit instructions is only used to indicate an indirect address.

All Indirect Addresses in the indirect address chain that are context relative are relative to the Data Base (Base #0) unless the indirect address is indexed. If the indirect address is indexed, the specified Base Indicant from the index register will be used to determine the Memory Area that contains the indirect address. If the indirect address is extended with a "D" and no index register is specified, the address is relative to the Code Base (Base #1).

An indexed branch address should resolve to a Base Indicant value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments.

3.2.4 ADDRESS INDEXING

The Index Register Format is defined as:

```
 SIGN  BI   D6   D5   D4   D3   D2   D1
+----+----+----+----+----+----+----+----+
| S  | I  | D  | D  | D  | D  | D  | D  |
+----+----+----+----+----+----+----+----+
```

S = Sign digit

I = Base Indicant

D = Decimal digit 0 thru 9

The address of the Index Register points to the sign digit. If indexing is specifed, the Index Register contents (D6 - D1) are added to or subtracted from the address depending on the value of the sign digit.

The value of the Base Indicant indicates which Base/Limit pair is associated with the indexed address. The Base Indicant must accompany the address in all further processing. The specified base is added to the sum of the address and the decimal field of the index register.

Some values of the Base Indicant may be invalid. See Appendix A - Compatibility Notes (A.32).

An attempt to index below the BASE or above the LIMIT (see Section 5.1) will cause an Address Error Fault (AEX = 11). See Appendix A - Compatibility Notes (A.30).

An undigit in the decimal field of an Index Register will cause an Address Error Fault (AEX = 12).

An indexed branch address should resolve to a Base Indicant value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments.

3.2.4 ADDRESS INDEXING (Continued)

Three Index Registers (IX1, IX2 & IX3) occupy a reserved area of memory, relative to Base #0. They are located at memory addresses 08, 16 and 24 respectively.

These Index Registers may also be loaded with the Load Index Register instruction (OP = 67) and stored with the Store Index Register instruction (OP = 68).

The following instructions and procedures have the ability to change the index registers that are maintained in reserved memory as an implicit operand:

OP	Mnemonic	Name	
30	BCT	Branch Communicate	(IX3)
31	NTR	Enter	(IX3)
32	EXT	Exit	(IX3)
35	VEN	Virtual Enter	(IX3)
37	SLL	Search Link List	(IX1)
38	SLD	Search Delink	(IX1,IX2)
39	SEA	Search	(IX1)
62	HCL	Hyper Call	(IX3)
63	RET	Return	(IX3)
64	SLT	Search List	(IX1,IX2)
66	STB	Search Table	(IX1)
91	SRD	Scan R/D	(IX1)
--	HCP	Hardware Call	(IX3)

3.2.4 ADDRESS INDEXING (Continued)

The four Mobile Index Registers (IX4, IX5, IX6 & IX7) may be loaded indivually or collectively with a Load Index Register instruction (OP = 67) and stored indivually or collectively with a Store Index Register instruction (OP = 68). The processor must maintain the value of these registers.

The following instructions and procedures set the Mobile Index registers invalid after storing them on a stack:

OP	Mnemonic	Name
30	BCT	Branch Communicate
62	HCL	Hyper Call
--	INP	Interrupt Procedure
--	HCP	Hardware Call Procedure

The following instructions restore the Mobile Index registers that were stored on a stack.

OP	Mnemonic	Name
63	RET	Return (HCL & HCP variant)
93	BRV	Virtual Branch Reinstatee

3.2.5 ADDRESS CONTROLLER

The two low-order bits of the first digit of the address field provide information that refers to the particular address or to the type of data stored at that address.

The two bits generally carry the following significance:

00	Unsigned 4-bit format (UN)
01	Signed 4-bit format (SN)
10	Unsigned 8-bit format (UA)
11	Indirect Address (IA)

Some combinations are prohibited in some instructions and may be variants in other instructions.

The values of the first digit for both the low (controller) and the high order (indexing) bits are shown below.

UN	SN	UA	IA			
0	1	2	3	No indexing		
4	5	6	7	IX1	IX4	IX7
8	9	A	B	IX2	IX5	
C	D	E	F	IX3	IX6	

Examples:

1. A-address = 601000.

   From the above table, 6 means UA and IX1. This means add 1000 and the contents of Index Register One to the value of the Base. The data at this address is processed as eight bit units.

2. A-address = F00000.

   From the table, F means IA and IX3. This means that the indirect address will be found by adding zero to the contents of IX3 and then adding the base.

3.3 OPERATOR CODE

The first two digits of an instruction are used to define the operation. All unassigned operator codes are reserved for expansion. The occurrence of an invalid operator code is an Invalid Instruction Fault (IEX = 01) and will cause a Hardware Call procedure to be executed.

3.4 FIELD LENGTH

The next four digits may be used to define field length. AF and BF generally refer to the data field length of the "A" operand and the "B" operand respectively. Maximum field length of 100 units is indicated when the two digit value is equal to 00.

3.4.1 INDIRECT FIELD LENGTH

Indirect Field Length is specified by setting the two high order bits (8 & 4) of the most significant digit True. When Indirect Field Length is specified, a two-digit memory location relative to the Data Base (Base #0), is addressed. The information at this address specifies the actual field length or another two digit memory location.

The relative address of the Indirect Field Length information is specified by the two low order bits of the most significant digit and the three high order bits of the least significant digit.

```
    MSD | LSD
   +-----+-----+
 8 |  I  |  U  |     I    = Indirect Field Length Flag = 1
   +-----+-----+
 4 |  I  +  U  |     T    = Tens Position of Address + Base
   +-----+-----+
 2 |  T  |  U  |     U    = Units Position of Address + Base
   +-----+-----+
 1 |  T  |  0  |     0    = MUST BE ZERO
   +-----+-----+
```

3.4.1 INDIRECT FIELD LENGTH (Continued)

Twenty even numbered indirect addresses ranging from 00 to 38, relative to Base #0, are available. The following table decodes the relative address of the indirect field length specified by AF or BF.

MSD	LSD 0	2	4	6	8
C	00	02	04	06	08
D	10	12	14	16	18
E	20	22	24	26	28
F	30	32	34	36	38

Example: If AF=C4 and Base #0 = 1000, the two digit length of the "A" data field or another indirect field length is found at absolute address 1004.

Only the following instructions have Indirect Field Length capability for both AF and BF:

OP	Mnemonic	Name
01	INC	Increment
02	ADD	Add
03	DEC	Decrement
04	SUB	Subtract
05	MPY	Multiply
06	DIV	Divide
08	MVD	Move Data
09	MVL	Move Links
10	MVA	Move Alpha
11	MVN	Move Numeric
12	MVW	Move Words
13	MVC	Move and Clear Words
14	MVR	Move Repeat
15	TRN	Translate

3.4.1 INDIRECT FIELD LENGTH (Continued)

OP	Mnemonic	Name
16	SDE	Scan-Equal
17	SDU	Scan-Unequal
18	SZE	Scan-Zone Equal
19	SZU	Scan-Zone Unequal
30	BCT	Communicate
31	NTR	Enter
35	VEN	Virtual Enter
37	SLL	Search Link List
38	SLD	Search Link Delink
39	SEA	Search
42	AND	And
43	ORR	Or
44	NOT	Not
45	CPA	Compare Alpha
46	CPN	Compare Numeric
49	EDT	Edit
60	LOK	Lock/Unlock
61	ASP	Adjust Stack Pointer
62	HCL	Hyper Call
64	SLT	Search List
65	WHR	Write Hardware Registers
66	STB	Search Table
67	LIX	Load Index Registers
68	SIX	Store Index Registers
85	CIO	Convert I/O
86	ATE	Alter Table Entry
87	MOP	Measurement OP
88	D2B	Decimal to Binary
89	B2D	Binary to Decimal
90	INT	Interrupt
91	SRD	Scan Result Descriptor
92	RAD	Read Address
94	IIO	Initiate I/O
95	RDT	Read Time of Day
97	STT	Set Time of Day
98	IOC	I/O Complete
99	SST	System Status
A0	MVS	Move Strings
A1	CPS	Compare Strings
A2	HSH	Hash Strings

3.4.1 INDIRECT FIELD LENGTH (Continued)

The following instructions have Indirect Field Length capability in AF only. BF indirect field length will be ignored.

OP	Mnemonic	Name
33	BRT	Bit Reset
34	BST	Bit Set
40	BZT	Bit Zero Test
41	BOT	Bit One Test
48	HBK	Halt Breakpoint *

* See Appendix A - Compatibility Notes (A.25).

3.4.2 LITERALS (AF ONLY)

The literal capability can only be specified by "AF". An "A" Field Literal is specified by setting the 8 and 2 bits of the most significant digit of AF True and the 4 bit False. The Literal flag indicates that the "A" address syllable of the instruction does not contain an address but does contain the literal data that is to be used by the instruction. The six digits of the "A" address portion of the instruction are the operand itself and not the address index, address controller or the operand address. The Literal is LEFT JUSTIFIED. Literal capability and AF Indirect Field Length cannot be specified at the same time.

```
     MSD | LSD
   +-----+-----+
 8 |  L  | A0  |    L = Literal Flag  = 1
   +-----+-----+
 4 |  0  |  U  |    0 = Must be Zero
   +-----+-----+
 2 |  L  |  U  |    U = Units of Literal Length
   +-----+-----+
 1 | A1  |  U  |    A = Address controller of Literal
   +-----+-----+        field (see following table)
```

3.4.2 LITERALS (Continued)

Bits A1 and A0 of "AF" indicate the literal data type.

A1	A0	Controller	Unit Length
0	0	Unsigned 4-bit (UN)	1-6 digits
0	1	Signed 4-bit (SN)	1-5 digits plus sign
1	0	8-bit (UA)	1-3 characters
1	1	Reserved	

The following improper useage of literal will produce an Invalid Instruction Fault (IEX = 22). See Appendix A - Compatibility Notes (A.28).

1. Controller = 3 (A1*A0)

2. Controller = 2 (A1*A0/) and length >3

3. Controller = 1 (A1/*A0) and length >5

4. Controller = 0 (A1/*A0/) and length >6

The data itself must be left-justified in the six digit field that would have contained the non-extended A address if it were not literal.

3.4.2 LITERALS (Continued)

Only the following instructions have literal capability:

OP	Mnemonic	Name
01	INC	Increment
02	ADD	Add
03	DEC	Decrement
04	SUB	Subtract
05	MPY	Multiply
06	DIV	Divide
10	MVA	Move Alpha
11	MVN	Move Numeric
14	MVR	Move Repeat
16	SDE	Scan-Equal
17	SDU	Scan-Unequal
18	SZE	Scan-Zone Equal
19	SZU	Scan-Zone Unequal
37	SLL	Search Link List
38	SLD	Search Link Delink
39	SEA	Search
*40	BZT	Bit Zero Test
*41	BOT	Bit One Test
42	AND	And
43	ORR	Or
44	NOT	Not
45	CPA	Compare Alpha
46	CPN	Compare Numeric
49	EDT	Edit
61	ASP	Aojust Stack Pointer
87	MOP	Measurement OP
88	D2B	Decimal to Binary
89	B2D	Binary to Decimal
94	IIO	Initiate I/O
98	IOC	I/O Complete

*Not recommended for general use

3.4.2 LITERALS (Continued)

The Literal Flag is invalid in the following instructions and will cause an Invalid Instruction Fault (IEX = 21):

OP	Mnemonic	Name
08	MVD	Move Data
09	MVL	Move Links
31	NTR	Enter
33	BRT	Bit Reset
34	BST	Bit Set
64	SLT	Search List
66	STB	Search Table
67	LIX	Load Index Registers
68	SIX	Store Index Registers
85	CIO	Convert I/O
90	INT	Interrupt
92	RAD	Read Address
93	BRV	Virtual Branch Reinstate
95	RDT	Read Time of Day
97	STT	Set Time of Day
99	SST	System Status
A0	MVS	Move Strings
A1	CPS	Compare Strings
A2	HSH	Hash Strings

The Literal Flag is ignored in the following instructions:

OP	Mnemonic	Name
47	SMF	Set Mode
48	HBK	Halt Breakpoint
63	RET	Return
84	ACM	Accumulator Manipulate
91	SRD	Scan Resust Descriptor

3.4.2 LITERALS (Continued)

The Virtual Enter (VEN) and Hypercall (HCL) instructions use concatenated AF/BF field lengths or an AF literal. In this case, an AF literal of B1, B2 or B3 will be interpreted as a length of 1, 2 or 3 characters in the "A" location. All other literals are not allowed and will cause an Invalid Instruction fault (IEX = 21).

3.5 INSTRUCTION OPERAND OVERLAP DEFINITIONS

3.5.1 TOTAL OVERLAP

Two operands totally overlap if their addresses, address controllers and field lengths are identical.

3.5.2 PARTIAL OVERLAP

Two operands partially overlap if at least one digit of each occupy the same memory location and at least one of the following conditions is true.

1) Starting operand addresses are not equal.
2) Data types are not equal.
3) Field lengths are not equal.

3.5.3 MATCHING TYPE-ADDRESS OVERLAP

A type of partial overlap where the operands addresses are the same and data types are the same, but the field lengths are not the same.

4 PROCESSOR STATE

4.1 INTERNAL PROCESSOR STATE

The V-Series architecture is basically a memory-to-memory computer architecture. However, some processor state is loaded from memory by the processor at specific times, and any succeeding modifications of those fields in main memory has no immediate effect upon the processor operation.

The following list specifies the global state set up in memory by software and loaded within the processor:

1. REINSTATE LIST ADDRESS
2. MEMORY AREA STATUS TABLE ADDRESS
3. SNAP PICTURE ADDRESS
4. MEMORY ERROR REPORT ADDRESS
5. ** KERNEL MEMORY AREA TABLE BASE/LIMIT ENTRIES
6. MCP ENVIRONMENT TABLE ADDRESS
7. NUMBER OF ENTRIES IN THE MCP ENVIRONMENT TABLE
8. TIME OF DAY

** Not all processors load this item of processor state. See Appendix A - Compatibility Notes (A.44).

4.1 INTERNAL PROCESSOR STATE (Continued)

The following list specifies the state maintained within the processor for the CURRENT TASK:

1. TASK REINSTATE LIST POINTER - points to entry for current task

2. TASK USER ENVIRONMENT TABLE POINTER - points to ET for current task

3. NUMBER OF ENTRIES IN TASK USER ENVIRONMENT TABLE

4. TASK USER SERVICES MEMORY AREA TABLE POINTER - points to USMAT for current task

5. NUMBER OF ENTRIES IN THE USER SERVICES MEMORY AREA TABLE

6. MCP DATA MEMORY AREA BASE/LIMIT - points to MCP DATA MA for current task

7. ACTIVE ENVIRONMENT NUMBER - active environment number for current task

8. PROGRAM COUNTER - program address for current task

9. TIME SLICE REMAINING

10. ACCUMULATOR

11. MEASUREMENT REGISTER

12. INTERRUPT MASK

13. MOBILE INDEX REGISTERS

14. COMPARISON TOGGLES (COMS)

15. OVERFLOW TOGGLE

16. MODE INDICATORS

4.2 OVERFLOW FLAG

The Overflow Flag indicates that the result field length of an arithmetic, Move Alphanumeric, or Move Numeric operation is not sufficient to store the result.

The Overflow Flag is not cleared at the beginning of an arithmetic operation, but is preserved. Therefore, it indicates overflow that has occurred any time before or during a series of arithmetic operations or other interspersed non-arithmetic operations.

The Overflow Flag is cleared by the Conditional Branch on Overflow (OP = 28), Edit (OP = 49) and Search (OP = 39) instructions.

The Overflow Flag is stored in memory and reset by the Branch Communicate (OP = 30), Enter (OP = 31), Virtual Enter (OP = 35), and Hyper Call (OP = 62) instructions. The Overflow Flag is also stored in memory and reset by the Interrupt and the Hardware Call procedures.

The Overflow Flag is unconditionally restored from memory by the Return (OP = 63) and Virtual Branch Reinstate (OP = 93) instructions and conditionally restored from memory by the Exit (OP = 32).

The COM and OVF Flags Field referenced elsewhere in this documentation contain the following information:

Information	Digit	Bit
Reserved	0*	3
Reserved		2
Reserved		1
Reserved		0
Reserved	1	3
Overflow Flag		2
Comparison Low Flag		1
Comparison High Flag		0

* Note: Digit 0 is used as a flag during the EXIT instruction to specify whether to restore the settings of the Overflow and Comparison flags.

4.2 OVERFLOW FLAG (Continued)

The Overflow Flag is affected by the following instructions and procedures:

OP	Mnemonic	Name
01	INC	Fixed Point Arithmetic
02	ADD	
03	DEC	
04	SUB	
06	DIV	
10	MVA	Move Alphanumeric
11	MVN	Move Numeric
28	OFL	Branch on Overflow
30	BCT	Branch Communicate
31	NTR	Enter
32	EXT	Exit
35	VEN	Virtual Enter
39	SEA	Search
49	EDT	Edit
50	IAD	Integer Arithmetic
51	IAS	
52	ISU	
53	ISS	
54	IMU	
55	IMS	
57	IMI	
62	HCL	Hyper Call
63	RET	Return
70	RAA	Real Arithmetic
71	RAS	
72	RSU	
73	RSS	
74	RMU	
75	RMS	
76	RDV	
77	RDS	
84	ACM	Accumulate Manipulate
88	D2B	Decimal to Binary Conversion
89	B2D	Binary to Decimal Conversion
93	BRV	Virtual Branch Reinstate
--	INP	Interrupt Procedure
--	HCP	Hardware Call Procedure

4.3 COMPARISON FLAGS (COMH, COML)

The states of the Comparison Flags are:

COML	COMH	
0	0	Null (Clear)
0	1	Greater or High
1	0	Less or Low
1	1	Equal

The Comparison Flags will be stored in memory and reset by the Branch Communicate (OP = 30), Enter (OP = 31), Virtual Enter (OP = 35), and Hyper Call (OP = 62) instructions. They are also stored in memory and reset by the Interrupt and the Hardware Call procedures.

The Comparison Flags are unconditionally restored from memory by the Return (OP = 63) and Virtual Branch Reinstate (OP = 93) instructions and conditionally restored from memory with by the Exit (OP = 32).

The layout of the COM Flags Field is displayed in Section 4.2.

4.3 COMPARISON FLAGS (Continued)

The Comparison Flags are affected by the following instructions and procedures:

OP	Mnemonic	Name
01	INC	Fixed Point Arithmetic
02	ADD	
03	DEC	
04	SUB	
05	MPY	
06	DIV	
10	MVA	Move Alphanumeric
11	MVN	Move Numeric
16	SDE	Scan to Delimiter - Equal
17	SDU	Scan to Delimiter - Unequal
18	SZE	Scan to Delimiter - Zone Equal
19	SZU	Scan to Delimiter - Zone Unequal
30	BCT	Branch Communicate
31	NTR	Enter
32	EXT	Exit
33	BRT	Bit Reset
34	BST	Bit Set
35	VEN	Virtual Enter
37	SLL	Search Link List
38	SLD	Search Link Delink
39	SEA	Search
40	BZT	Bit Zero Test
41	BOT	Bit One Test
42	AND	And
43	OR	Or
44	NOT	Not
45	CPA	Compare Alphanumeric
46	CPN	Compare Numeric
49	EDT	Edit
50	IAD	Integer Arithmetic
51	IAS	
52	ISU	
53	ISS	
54	IMU	
55	IMS	
57	IMI	
58	ILD	
59	IST	
60	LOK	Lock/Unlock

4.3 COMPARISON FLAGS (Continued)

OP	Mnemonic	Name
62	HCL	Hyper Call
63	RET	Return
64	SLT	Search List
66	STB	Search Table
70	RAA	Real Arithmetic
71	RAS	
72	RSU	
73	RSS	
74	RMU	
75	RMS	
76	RDV	
77	RDS	
78	RLD	
79	RST	
84	ACM	
88	D2B	Decimal to Binary
89	B2D	Binary to Decimal
91	SRD	Scan Result Descriptor
92	RAD	Read Address Register
93	BRV	Virtual Branch Reinstate
94	IIO	Initiate I/O
A1	CPS	Compare Strings
--	INP	Interrupt Procedure
--	HCP	Hardware Call Procedure

The Comparison Flags are not altered by conditional or unconditional branching.

4.4 MODE INDICATORS

The Mode Indicators are used to specify the operating mode of the processor.

The Mode Indicators are stored in memory in the following format:

INFORMATION	DIGIT	BIT
Reserved	0	3
Reserved	0	2
Reserved	0	1
Reserved	0	0
Soft Fault Enable	1	3
Privileged/User Mode	1	2
Trace Mode	1	1
Snap Enable	1	0

The Mode Indicators may be changed by the following instructions and procedures:

OP	Mnemonic	Name
30	BCT	Branch Communicate
62	HCL	Hyper Call
63	RET	Return
93	BRV	Virtual Branch Reinstate
--	INP	Interrupt Procedure
--	HCP	Hardware Call Procedure

4.4.1 SNAP ENABLE INDICATOR

The Snap Enable Indicator must be set in addition to the Snap Picture Enable (See Sec. 4.6) to allow a Snap picture to be taken and stored at the Snap Picture address that has been set with the Write Hardware Register instruction (OP = 65:BF = 01).

4.4.2 TRACE INDICATOR

The Trace Indicator enables the Trace function of the processor. The Trace Indicator is either set or reset by the Hyper Call (OP = 62), the Return (OP = 63), or the Virtual Branch Reinstate (OP = 93) instructions or the Hardware Call procedure. The Trace Indicator is always reset by the Interrupt Procedure.

When the Trace function is enabled, a Trace Fault Hardware Call procedure will be executed at the completion of the current instruction except after the Hardware Call Return variant of the Return (OP = 63) instruction. The Hardware Call procedure will store the instruction address of the next instruction to be executed along with some trace parameters on the stack. See Appendix A - Compatibility Notes (A.50).

The Hardware Call Return variant of the Return (OP = 63) instruction will prevent a Hardware Call procedure due to the Trace Indicator until the next instruction has been executed.

4.4.3 PRIVILEGED/USER MODE INDICATOR

The Privileged/User Mode Indicator is used to control the use of certain instructions that may only be executed by the operating system. The system is in Privileged Mode, which allows all instructions to be executed, if the indicator is set. The system is in User Mode if the indicator is reset. An attempt to use a privileged instruction while in User Mode will cause an Invalid Instruction fault (IEX = 02).

The following instructions may only be executed in Privileged Mode:

OP	Mnemonic	Name
60	LOK	Lock
63	RET	Return (HCL & HCP variant)
65	WHR	Write Hardware Register
85	CIO	Convert I/O
86	ATE	Alter Table Entry
90	INT	Interrupt
91	SRD	Scan Result Descriptor
92	RAD	Read Address
93	BRV	Virtual Branch Reinstate
94	IIO	Initiate I/O
97	STT	Set Time-of-Day Timer
98	IOC	I/O Complete
99	SST	System Status

The Privileged/User Mode Indicator also restricts the use of the MCP Environment Number in certain instructions. In the Virtual Enter (OP = 35), the Virtual Exit variant of Return (OP = 63), the Move String (OP = A0), the Compare String (OP = A1) and the Hash String (OP = A2) instructions, if the most significant digit of the Environment Number is equal to a "D" and the system is not in Privileged Mode cause an Invalid Instruction fault (IEX = 31). In the Move String (OP = A0), Compare String (OP = A1) and Hash String (OP = A2) instructions, if the Environment Number is equal to "zero" and the system is not in Privileged Mode cause an Invalid Instruction fault (IEX = 32).

4.4.4 SOFT FAULT ENABLE INDICATOR

The Soft Fault Enable Indicator enables the soft fault reporting feature of the V-Series operating system.

When this indicator is set the Branch Communicate (OP = 30), Hyper Call (OP = 62), Return (OP = 63) and Virtual Branch Reinstate (OP = 93) instructions will examine the Soft Fault Pending Flag in the Reinstate List entry for the current task, and cause a Hardware Call procedure to be executed if the digit is not equal to zero.

4.5 MEMORY ERROR REPORT ENABLE

The Memory Error Report Enable allows one memory error report to be written into memory.

This indicator affects the contents of main memory specified by the value of the Memory Error Report Address that has been set by the Write Hardware Register instruction (OP = 65:BF = 02).

This enable is set by the Write Hardware Register instruction (OP = 65:BF = 02) and is turned off automatically after the report has been written into memory.

4.6 SNAP PICTURE ENABLE

The Snap Picture Enable allows one SNAP picture to be stored in memory.

This enable affects the contents of main memory specified by the value of the Snap Picture Address that has been set by the Write Hardware Register instruction (OP = 65:BF = 01).

This indicator is set by the Write Hardware Register instruction (OP = 65:BF = C1) and is turned off automatically after the picture has been written into memory.

4.7 TASK TIMER

The Task Timer is a counter that is used to interrupt a task when its time slice has ended. The maximum timer value is about 100 seconds. The most significant digit of the timer controls the timer interrupt.

If the Timer Interrupt bit is set in the Interrupt Mask, an Interrupt procedure occurs, that stores the address of the next instruction to be executed, whenever the most significant digit of the timer is equal to zero.

If the Timer Interrupt bit is not set in the Interrupt Mask, the timer will continue to decrement at the same rate until an Interrupt procedure is executed for any reason. At that time the value of the Task Timer will be stored in the Reinstate List Entry for the interrupted task and the Task Timer set to its maximum value.

If an interrupt has not occurred before the entire task timer reaches zero, a task timer fault will occur, which will cause a hardware call that will store the address of the next instruction to be executed. See Appendix A - Compatibility Notes (A.45).

The Task Timer is affected by the Virtual Branch Reinstate instruction (OP = 93) and the Interrupt procedure.

The resolution of the Task Timer is machine dependent. See Appendix A - Compatibility Notes (A.36).

4.8 KERNEL MODE

Kernel Mode is the special mode the processor is in while executing the MCP Kernel routine. The MCP Kernel is entered with the execution of the Interrupt procedure and is exited with the execution of the Virtual Branch Reinstate instruction (OP = 93).

An attempt to execute a Hardware Call procedure while in Kernel Mode will cause the system to REDLIGHT halt after storing the fault indicators in absolute memory location 72 - 81.

When a REDLIGHT halt occurs, a SNAP picture is taken (if enabled) and the processor stops. The processor cannot continue from this point as the error that caused the REDLIGHT is in a non-recoverable portion of the system. The error must be cured and the system reinitialized manually.

5 TASK ADDRESSING ENVIRONMENT

5.1 TASK ADDRESSING CAPABILITIES

A task may have up to eight different areas of main memory addressable in its local environment at any one time. These local memory areas are located using the following data structures: the Reinstate List, the MCP Environment Table, the User Environment Table, and Memory Area Tables. These tables are described in more detail in later sections. However, the basic linkages between these tables are described briefly below.

Each task has an entry in a system array called the Reinstate List in which the processor and MCP maintain information about that task.

A task may execute instructions from MCP code modules or user program code. All of the code modules for the MCP have entries in the MCP Environment Table. Entry #0 of the Reinstate List contains a pointer to the MCP Environment Table. All of the code modules for a user program have entries in the User Environment Table for that task. The Reinstate List entry for each task contains a pointer to the User Environment Table for that task.

5.1 TASK ADDRESSING CAPABILITIES (Continued)

Each MCP or User Environment Table entry points to a Memory Area Table which describes the local addressing environment for that code. The first eight entries in the Memory Area Table contain the actual base/limit pairs for each memory area or indirect pointers to the actual base/limit pairs.

The smallest allocatable unit of memory is a Memory Area. It may be from 1,000 to 1,000,000 digits in size in increments of 1000 digits.

The following instructions or procedures change the operating environment of the processor by resolving the first eight entries in the specified Memory Area Table and retaining the Base and Limit values for memory access protection. See Appendix A - Compatibility Notes (A.34).

OP	Mnemonic	Name
30	BCT	Branch Communicate
35	VEN	Virtual Enter
62	HCL	Hyper Call
63	RET	Return
86	ATE	Alter Table Entry
93	BRV	Virtual Branch Reinstate
--	INP	Interrupt Procedure
--	HCP	Hardware Call Procedure

The processor has the ability to reference these eight Memory Areas as the local environment at any point in time (See Section 3.2). The processor also has the ability to reference non-local Memory Areas and provide memory access protection for such Memory Areas as are specified by the Convert I/O (OP = 85), the Alter Table Entry (OP = 86), the Move String (OP = A0), the Compare String (OP = A1), or the Hash String (OP = A2) instructions.

5.2 REINSTATE LIST

The Reinstate List is a system array set up by the MCP to control task switching on the processor. Every task has an entry in this table.

The Reinstate List is located in memory with a Write Hardware Register instruction (OP = 65:BF = 00).

Two of the entries have special significance. The entry for task #0 is not assigned a task, because its Environment Table Address field contains the address of the MCP Environment Table. Task #1 is reserved for the MCP Kernel code.

See Appendix A - Compatibility Notes (A.46) for a detailed layout of the entry.

5.3 ENVIRONMENT TABLE

The Environment Tables are system arrays set up by the MCP to inform the processor of the legal addressing environments for a task. A task has access to addressing environments in the MCP Environment Table or that task's USER Environment Table.

The MCP Environment Table is located by the Environment Table Address field in the Reinstate List entry for Task #0. This Environment Table is shared by all tasks.

Every task has its own USER Environment Table, located by the Environment Table Address field in the Reinstate List entry for that task.

See Appendix A - Compatibility Notes (A.48) for a detailed layout of the Environment Table entry.

5.4 MEMORY AREA TABLE

A Memory Area Table (MAT) contains 1-100 entries which contain the actual Base/Limit pairs for a memory area or point indirectly via Copy descriptors to the actual Base/Limit pairs. It may be an executable or non-executable Memory Area Table.

Each environmental table entry of a task contains the address of a Memory Area Table and the number of entries in that table. Each task when running has one of its Memory Area Tables loaded into the hardware. This is called its local addressing environment. A task can have up to eight different memory areas in its local addressing environment at any one time. Its local addressing environment is defined by an executable Memory Area Table. An executable Memory Area Table contains eight entries, of which Base/Limit #1 references a code memory area (i.e., instructions are fetched from this memory area).

A non-executable Memory Area Table does not contain a reference to a code memory area in Base/Limit pair #1. It is used for storage of Memory Area descriptors (Base/Limit pairs or Copy descriptors to Base/Limit pairs).

Every task has a User Services Memory Area Table (USMAT). This non-executable Memory Area Table describes MCP memory areas containing privileged information about that task. To protect these memory areas from user access, this Memory Area Table is located by the Memory Area Table Address field in the USER Environment Table entry #0 for the task. Thus to access any of these memory areas via an instruction, the user must specify an Environment Number of zero, which is illegal except in Privileged Mode.

A Memory Area Table entry can have the following Memory Area Table entries: Original Entry, "C" Copy Entry, "E" Copy Entry, Memory Area Fault Entry, and Unused Entry. See Appendix A - Compatibility Notes (A.34) for a detailed layout of each entry.

Some processors require an ATE to notify it when a MAT entry is modified. See Section 20 on the ATE instruction for details.

5.5 MEMORY AREAS

The smallest allocatable unit of memory is a Memory Area. It may be from 1,000 to 1,000,000 digits in size in increments of 1000 digits. For example, a software code module that is 57,244 digits in size will be assigned to a 58,000 digit Memory Area. A software code module of only 150 digits will be assigned to a 1,000 digit Memory Area.

A memory area is located by a Base/Limit pair contained in a Memory Area Table. For references to data or code in the local addressing environment, the base relative memory addresses are added to the selected Base value determine the absolute memory location.

Memory access protection is provided by comparisons of the Base and Limit Values to the address of the requested memory access to insure that the value of the requested address is less than the Limit value but greater than or equal to the Base value. If the address of requested memory access is outside of the specified Memory Area, cause an Address Error fault (AEX = 20-26) and terminate the instruction without storing any further data into memory.

5.6 MEMORY AREA STATUS TABLE (MAST)

The Memory Area Status Table is located in memory with a Write Hardware Register instruction (OP = 65:BF = 03). Each entry in the Memory Area Status Table contains information that is related to a Memory Area that is known to the system.

See Appendix A - Compatibility Notes (A.47) for a detailed layout of the entry.

5.7 LOCATING A MEMORY AREA TABLE ENTRY

A Memory Area Table Entry is specified by a Task Number (TN), Environment Number (EN), and a Memory Area Number (MAN).

To locate a Memory Area Table (MAT) or an entry within a MAT, one must traverse the address links through the Reinstate List, the MCP Environment Table or Task User Environment Table, and Memory Area Tables.

The Task Number represents an array subscript into the Reinstate List of 0000 to 9999. (The actual number of possible tasks may be limited by memory constraints.) The address of the Reinstate List is provided by software with a Write Hardware Register instruction (BF = 00). The processor maintains an internal pointer to the Reinstate List entry for the current task, but must recalculate any references to Reinstate List entries for other tasks. The Reinstate List entry for Task #0 contains the MCP Environment Table Address and that table's number of entries rather than information about an actual independent task.

If the first digit of the EN is equal to a "D", then the five least significant digits represent an array subscript into the MCP Environment Table of 00000 to 99999. The Reinstate List entry for Task #0 contains the address of the MCP Environment Table in its Environment Table Address field and the size of the MCP Environment Table in its Number of Entries in the Environment Table Field.

If the first digit of the EN is equal to "0 - 9", this six digit number represents an array subscript into a User Environment Table of 000000 to 999999. (The actual number of environments for a task may be limited by memory constraints.) The Reinstate List entry for a task contains the address of its task's User Environment Table in its Environment Table Address field and the size of its task's User Environment Table in its Number of Entries in the Environment Table field.

5.7 LOCATING A MEMORY AREA TABLE ENTRY (Continued)

If the first digit of the EN is equal to any other value, then cause an Address Error fault (AEX = 52 if the EN came from an Environment Descriptor or AEX = 62 if the EN came from a MAT entry) and terminate the instruction with no further action. If undigits exist in the last five digits of the EN, cause an Address Error fault (AEX = 53 if the EN came from an Environment Descriptor or AEX = 63 if the EN came from a MAT entry) and terminate the instruction with no further action.

If the array subscript portion of the Environment Number is larger than the number in the Number of Entries field for the relevant Environment Table, cause an Address Error fault (AEX = 57 if the EN came from an Environment Descriptor or AEX = 67 if the EN came from a MAT entry) and terminate the instruction with no further action.

If a MAT is being located by an Environment Number contained in user addressable memory (i.e., other than a MAT Copy Type entry or an MCP Function Table entry), then the following additional security check is done:

> If the processor is not in Privileged Mode and either the Environment Number equals zero (excluding the special cases of VIRTUAL ENTER and RETURN (VIRTUAL EXIT Variant)) or the first digit of the Environment Number equals "D", then cause an Invalid Instruction fault (AEX = 31) and terminate the instruction with no further action.

The Environment Table entry located above contains the address of the desired Memory Area Table in its Memory Area Table Address field and the size of the MAT in its Number of Entries in the Memory Area Table field. If the Memory Area Number parameter is greater than the value in the Number of Entries in the Memory Area Table field, then cause an Address Error Fault (AEX = 58 if the EN came from an Environment Descriptor or AEX = 68 if the EN came from a MAT entry). If any of the digits of the Memory Area Number contain undigits, then cause an Address Error Fault (AEX = 54 if the EN came from an Environment Descriptor or AEX = 64 if the EN came from a MAT entry).

5.8 RESOLVING A MEMORY AREA TABLE ENTRY

A Memory Area Table (MAT) entry may contain the following types of entries: an Original Type (the actual Base/Limit of the corresponding Memory Area), an Unused Type (entry is unused), "C" or "E" Copy Types (entry contains a "C" or "E" Copy Descriptor pointing to another MAT entry), or Memory Area Fault Type (the Memory Area is not currrently in main memory).

The algorithm for resolving a MAT entry is generally handled as described below. However, in some instruction algorithms, additional entry type checking is performed which may cause additional error faults. Refer to the particular instruction algorithms for these exceptions.

Copy Type entries provide levels of indirection for MAT entries, so that only one Original Entry may exist for a memory area. One Original Entry may have many Copy Type entries pointing to it. If the MAT entry being resolved is a Copy Type entry, then the information in this entry is used to locate another MAT entry which must then be resolved. If a chain of Copy Type entries exists, then this process will repeat itself until a non-Copy Type entry is found. The handling of this final non-Copy Type entry is described later in this section. The next paragraph explains how to determine the Copy Descriptor parameters needed to locate the next MAT entry in the chain.

If the MAT entry is a "C" Copy Type entry, then the Environment Number and Memory Area Number contained in the MAT entry (along with the current Task Number) are used to locate the next MAT entry to be resolved (See Section 5.7 for details). Since the information in a "C" Copy Type entry may only point to a MCP MAT or a MAT for the current task, it is frequently used by the MCP to gain access to memory areas belonging to whichever task it is currently servicing.

5.8 RESOLVING A MEMORY AREA TABLE ENTRY (Continued)

If the MAT entry is an "E" Copy Type entry, then the absolute address of the next MAT entry to be resolved is contained in the MAT entry. However, an "E" Copy Type entry must not point to another Copy Type entry or an Address Error fault (AEX = 69) will occur. The "E" Copy Type entry is typically used when a task wants to gain access to data in a memory area which belongs to antoher task. In this case, the owning task has the Original Entry for the shared memory area in its Memory Area Table and the other tasks point to it via an "E" Copy Type entry.

If the final MAT entry is an Original Type, then the Base/Limit pair contained in this entry is loaded directly into proper processor Base/Limit registers.

If the final MAT entry is a Memory Area Fault Type, then a Hardware Call Procedure reporting either a Soft Memory Area Fault or Hard Memory Area Fault is performed. A Soft Memory Area Fault is performed unless the particular instruction algorithm states otherwise. Note that an attempt to execute a Hardware Call Procedure in Kernel Mode causes a REDLIGHT halt (See Section 4.8).

If the final MAT entry is an Unused Type, then the Base and Limit registers for this entry are set equal to each other. However, since these Base/Limit values are invalid (i.e., Base/Limit = 000/000), any attempt to access memory via an Unused Entry will cause an Address Error fault (AEX = 2n).

5.9 LOADING A MEMORY AREA TABLE

Loading a Memory Area Table (MAT) consists of locating an eight entry executable MAT, resolving entries 0-7, and, if no error faults were detected, loading the final results into the corresponding processor Base/Limit registers 0-7.

This procedure is performed when a task changes or reloads its local environment during instructions like Hypercall, Branch Communicate, Virtual Enter (non-local variant), Return (Virtual Exit, Hyper Return, and Hardware Return variants), Interrupt, Virtual Branch Reinstate, and Alter Table Entry. It is also used in the Hardware Call Procedure and Interrupt processing.

5.10 MCP DATA AREA

Every task has a memory area (located by User Environment Number 0, Memory Area Number 0) called the MCP Data Area in which the MCP maintains information concerning that task. In addition, it contains the "reserved memory" of the MCP routines for that task (See Section 22.2 for a description of the MCP Data Area reserved memory). The MCP Data Area is described by Base/Limit pair #0 when a task is executing MCP code.

6 INTERRUPT PROCESSING

There are two mechanisms for interrupting an instruction stream to start executing the appropriate MCP routines: Interrupt Procedures and Hardware Call Procedures.

1. An Interrupt Procedure is executed as a response to certain instruction interrupts or maskable interrupts to transfer control of the processor to the MCP Kernel routine.

   An Interrupt Mask is used to indicate which of the maskable conditions will be allowed to interrupt the current processing. If any interrupt conditions have occurred, and the corresponding bit(s) in the Interrupt Mask is set, then an interrupt procedure is executed. The interrupt condition will be reset during the execution of the Interrupt procedure if the corresponding condition is set in the Interrupt Mask.

   The Interrupt Descriptor is as follows:

CONDITION	BIT		CAUSE
Reserved	7		
Instruction	6	-	Instruction-related Interrupt
Overtemp	5	-	System Overtemperature
Task Timer	4	-	MSD = 0
Reserved	3		
REAL TIME I/O	2	-	I/O COMPLETE Real Time Device.
NORMAL I/O ERROR	1	-	I/O COMPLETE, Exceptions, Non-Real Time Device.
NORMAL I/O	0	-	I/O COMPLETE, No Exceptions, Non-Real Time Device.

   The Instruction Interrupt is not maskable. However, the other Interrupt Descriptor bits are the logical "and" of the pending interrupt conditions and the Interrupt Mask.

6 INTERRUPT PROCESSING (Continued)

The definition of the eight bit Interrupt Mask is as follows:

CONDITION	BIT		CAUSE
Reserved	7		
Reserved	6		
Overtemp	5	-	System Overtemperature
Task Timer	4	-	MSD = 0
Reserved	3		
REAL TIME I/O	2	-	I/O COMPLETE Real Time Device.
I/O ERROR	1	-	I/O COMPLETE, Exceptions, Non-Real Time Device.
NORMAL I/O	0	-	I/O COMPLETE, No Exceptions, Non-Real Time Device.

The interrupt conditions are tested and the interrupt procedure initiated at the end of the current instruction with the address of the next instruction to be executed stored in the Interrupt Frame unless an instruction variant specifies that the address of the instruction that caused the interrupt is to be stored in the Interrupt Frame.

Instruction Interrupts are flagged in the Interrupt Descriptor with the actual Instruction Interrupt condition stored in the Instruction Interrupt Cause Descriptor located in absolute memory locations 32 - 33.

6 INTERRUPT PROCESSING (Continued)

The Instruction Interrupt Cause Descriptor is defined as follows:

VALUE	CAUSE
08-FF	Reserved
07	Failed Virtual Branch Reinstate
06	Executed Interrupt Instruction (OP = 90)
05	Failed Hardware Call
04	Released Event
03	Released Lock
02	Failed Event
01	Failed Lock
00	Reserved

An Interrupt caused by a failed Hardware Call procedure may store inconsistent values depending upon the cause of the failure.

An over-temperature condition in the system will cause an Interrupt procedure to be executed. After detection of the condition a processor-dependent time delay occurs before a system power-off is initiated.

If an Interrupt condition and a Hardware Call condition exist at the same time, the following algorithm is applied:

a. If an Instruction Interrupt condition and an instruction-related Hardware Call "error" condition (Invalid Arithmetic Data, Instruction Timeout, Address Error, Uncorrectable Memory Parity Error, Invalid Instruction) occurs at the same time, then the Instruction Interrupt condition is ignored as the instruction did not actually execute correctly.

b. If the Instruction Interrupt condition is a Failed Lock and the only Hardware Call condition is a Trace Fault, then the Trace Fault condition is ignored.

6 INTERRUPT PROCESSING (Continued)

c. If any other Fault conditions still remain, then a Hardware Call Procedure is performed.

d. If any Instruction Interrupt conditions remain or any maskable interrupt conditions are not masked by the Interrupt Mask (which may have just been loaded by a Hardware Call procedure), then an Interrupt Procedure is performed.

2. A Hardware Call procedure is executed as a response to certain processor detected faults. A Hardware Call procedure changes the system environment and transfers control to a software error handling routine.

The Hardware Call procedure stores the "state" of the processor in a Hardware Call Stack Frame on the stack that is associated with the called routine.

The instruction address of the failing instruction is included in the "state" that is stored as a result of the following faults: Address Error; Invalid Instruction; Invalid Arithmetic Data; Invalid Alter Table Entry; Accumulator Trap; Uncorrectable Memory Errors; Instruction Timeout; and certain Memory Area Faults.

The instruction address of the next instruction to be executed (unless one of the faults that require that the address of the failing instruction be stored is also present) is included in the "state" that is stored as a result of the following faults: Trace; Programatic Soft Fault; and other Memory Area Faults.

See the introductory discussion in Section 6 about Interrupt Procedures for a description of the algorithm to be applied when an Interrupt condition and a Hardware Call condition exist at the same time.

6.1 INTERRUPT PROCEDURE (INP)

The Interrupt procedure is used by the processor hardware to transfer the system environment to the MCP Kernel.

This procedure is used to enter the MCP Kernel function specified by the six digit address located at memory address 94, relative to the MCP Data Area for Task #1. Processor registers and "state" are stored in the Interrupt Frame and control is transferred to the MCP Kernel environment.

The following operations are peformed by this instruction to enter the MCP Kernel environment:

1. Store the current value of the Task Timer into the Time Slice Remaining Field of the Reinstate List Entry for the current Task. Set the value of the Task Timer to its maximum numeric value.

2. Store the two-digit Interrupt Descriptor into absolute memory locations 21-22.

3. Store the machine "state" of the interrupted task into the Interrupt Frame, located in the Reinstate List Entry for the interrupted task.

4. Selectively reset the interrupt conditions according to the Interrupt Mask. If the bit in the mask is equal to a "one", reset the corresponding condition. If the bit in the mask is equal to "zero" the corresponding condition will not be changed. Reset the Instruction Interrupt condition.

5. Set the machine "state" as follows:

INFORMATION	SET TO
Kernel Mode	SET
Active Environment Number	000000
Current Task Number	0001
Privileged/User Mode	PRIVILEGED
Trace Mode	NON-TRACING
Snap Enable	DISABLED
Soft Fault Enable	DISABLED
Measurement Register	0000 0000
Comparison & Overflow Flags	RESET

6.1 INTERRUPT PROCEDURE (Continued)

6. Set the MOPOK signal to "0" while the Measurement register is being changed and set it to a "1" at all other times.

7. Set the Interrupt Mask register to zero.

8. Store the four digit Task number for Task number one (0001) at absolute memory address 82.

9. Locate and load the Kernel Memory Area Table (MAT), which is the MAT pointed to by the first entry in the USER Environment Table for Task #1.

10. Execute an unconditional branch to the six-digit address, relative to Base #1, located at memory address 94, relative to Base #0. If any Hardware Call conditions exist, cause a REDLIGHT halt (See Section 4.6) after storing the fault indicators in absolute memory location 72 - 81.

6.2 HARDWARE CALL PROCEDURE (HCP)

The Hardware Call procedure is used by the processor hardware, when one of the specified faults exist, to enter the routine specified by the Hardware Call Function. The fault indicators are storea in a fixed memory location relative to Base #0 of the called function. Processor "state" is stored in a Hardware Call Stack Frame on a stack that is relative to Base #0 of the specified function. Control is transfered to the Hardware Call Procedure code.

If the processor is in the MCP Kernel environment, an attempt to execute a Hardware Call procedure will cause the processor to REDLIGHT halt (See Section 4.8) after the fault indicators have been stored in absolute memory location 72 - 81.

The following operations are performed by this procedure:

1. Locate the six digit address, relative to the MCP Data Area, of the Hardware Call Function entry in the MCP Function Table at memory address 87 relative to the MCP Data Area.

   The Hardware Call Function entry contains the following information:

INFORMATION	DIGITS
Environment Number	00-05
Next Instruction Address	06-11
Protection Field (DD)	12-13
Reserved	14-15
Interrupt Mask	16-17
Mode Indicators	18-19

   Note - The lowest memory address = 00

6.2 HARDWARE CALL PROCEDURE (Continued)

If the Protection Field is not equal to "DD", then this Hardware Call Procedure has failed. Perform the following steps:

a. Write the Fault Indicators into the Failed Hardware Call R/D Area Field of the Reinstate List entry for this task.

b. Write "05" into the State Indicator field of the same Reinstate List entry.

c. Perform an Interrupt Procedure, reporting a "05" (Failed Hardware Call) in the Instruction Interrupt Cause Description.

2. Load the Memory Area Table specified by the Environment Number contained in the Hardware Call Function Table entry.

If an invalid Environment Number or Memory Area Number is encountered or a Memory Area Fault is found during the loading of the MAT, then this Hardware Call Procedure has failed. Perform the following steps:

a. Write the Fault Indicators into the Failed Hardware Call R/D Area Field of the Reinstate List entry for this task.

b. Write "05" into the State Indicator field of the same Reinstate List entry.

c. Perform an Interrupt Procedure, reporting a "05" (Failed Hardware Call) in the Instruction Interrupt Cause Description.

6.2 HARDWARE CALL PROCEDURE (Continued)

3. Store ten digits of Fault Indicators into memory Location 72 - 81, relative to the newly loaded Base #0. The Fault Indicators contain the following information:

INFORMATION	DIGIT	BIT
Hard Memory Area Fault	72	3
Trace	72	2
Invalid Arithmetic Data (Undigits)	72	1
Soft Memory Area Fault	72	0
Invalid Instruction	73	3
Uncorrectable Memory Parity Error	73	2
Address Error	73	1
Instruction Timeout	73	0
Stack Overflow	74	3
Accumulator Trap	74	2
Snap Picture Taken	74	1
Soft Fault	74	0
Reserved	75	3
Reserved	75	2
Reserved	75	1
Task Timer Fault	75	0
Reserved	76-77	ALL
Address Error Extension	78-79	ALL
Invalid Instruction Extension	80-81	ALL

4. Reset the Fault Condition Indicators.

6.2 HARDWARE CALL PROCEDURE (Continued)

5. Use the stack pointer, located at address 40 (Relative to the newly loaded Base #0) as the starting address, relative to Base #0, to store the Hardware Call Stack Frame.

   The Hardware Call Stack Frame is stored on the stack in the following sequence.

	INFORMATION	DIGITS
Old TOS ==>	Accumulator	00-27
	Measurement Register	28-35
	Interrupt Mask	36-37
	Mobile Index Registers	38-69
	Mode Indicators	70-71
	COM and OVF Flags	72-73
	Active Environment Number	74-79
New IX3 ==>	Instruction Address *	80-85
	Saved IX3 Value	86-93
	Stack Frame Indicator (FD)	94-95
	Fault Environment Number **	96-101
	Fault Memory Area Number **	102-103
	Fault Task Number **	104-107
	Trace Information **	108-187
New TOS ==>		

Note - The lowest memory address = 00.

* Either store the address of the failing instruction or the address of the next instruction to be executed on the stack depending upon the type of fault.

** If the Hardware Call was caused by a Hard or Soft Memory Area fault, store the Environment Number, Memory Area Number and Task Number of the faulted entry as parameters on the stack. See Appendix A - Compatibility Notes (A.51). If in Trace Mode, the trace parameters are passed on the stack in the next 80 digits. See Appendix A - Compatibility Notes (A.50). The space for these parameters is always allocated whether or not they are actually present.

6.2 HARDWARE CALL PROCEDURE (Continued)

6. Store the new value of the next available stack location (Relative to the newly loaded Base #0) into memory address 40 (Relative to the newly loaded Base #0).

7. Set the two most significant digits of IX3 to "C0" and set the six least significant digits of IX3 to the initial address specified in memory address 40 (Relative to the newly loaded Base #0) plus 80.

   IX3 now points to the Next Instruction Address in the Stack Frame.

8. Set the machine "state" as follows:

INFORMATION	SET TO
Next Instruction Aodress	Function Table
Active Environment Number	Function Table
Interrupt Mask	Function Table
Mode Indicators	Function Table
Measurement Register (user field)	000000
Comparison & Overflow Flags	RESET
Trace Mode	NON-TRACING

9. Set the MOPOK line to "zero" while the Measurement register is being changed and set it to a "one" at all other times.

6.2 HARDWARE CALL PROCEDURE (Continued)

10. Using the new Base/Limit environment, resolve the next instruction address, relative to Base #1 and execute an unconditional branch to that address.

The use of the Mobile Index Registers or the Accumulator to pass parameters is invalid. The contents are not guaranteed.

The Hardware Call procedure fails if any faults exist at the completion of the Hardware Call procedure. If the Hardware Call procedure fails, store:

(a) the Fault indicators in the Failed Hardware Call R/D Area Field of the Reinstate List Entry for this task

(b) "05" into the State Indicator Field of the Reinstate List entry for this task

(c) "05" into absolute memory location 32 and cause an Instruction Interrupt to the MCP Kernel. The Interrupt procedure may store inconsistent values depending upon the cause of the failure.

6.2.1 FAULT INDICATORS

The Fault Indicators are described in the following paragraphs.

6.2.1.1 (DIGIT 72 BIT 3) - HARD MEMORY AREA FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because a Memory Area Fault entry was detected while loading a Memory Area Table in certain instructions [Hyper Call (OP = 62), Convert I/O (OP = 85), Move String (OP = A0), Compare String (OP = A1), and Hash String (OP = A2)] that require the faulted entry in order to execute the instruction. The Environment Number and Memory Area Number that point to this Memory Area Table entry were also stored as stack parameters.

6.2.1.2 (DIGIT 72 BIT 2) - TRACE FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the next instruction to be executed because the system is operating in Trace Mode.

The following information is passed on the stack on each trace fault. See Appendix A - Compatibility Notes (A.50).

1. Program address of previous instruction (the one being traced).
2. Program address of next instruction.
3. Opcode.
4. Resolved AF with an indirect flag indication.
5. Resolved BF ~~with~~ an indirect flag indication.
6. A address, resolved (including index register used and address controller).
7. B address, resolved (including index register used and address controller).

6.2.1.2 TRACE FAULT (Continued)

8. C address, resolved (including index register used and address controller).

9. Overflow and Comparison flags.

6.2.1.3 (DIGIT 72 BIT 1) - INVALID ARITHMETIC DATA FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an undigit other than the sign digit has been detected in an arithmetic operand. See Appendix A - Compatibility Notes (A.16).

A SNAP picture will be taken if enabled by the Snap Enable Indicator (Sec. 4.4.1) and the Snap Picture Enable (Sec. 4.6).

6.2.1.4 (DIGIT 72 BIT 0) - SOFT MEMORY AREA FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the next instruction to be executed because a Memory Area Fault entry was detected while loading a Memory Area Table in those instructions that do not require the faulted entry in order to execute the instruction. The Environment Number, Memory Area Number and Task Number that point to this Memory Area Table entry were also stored as stack parameters.

6.2.1.5 (DIGIT 73 BIT 3) - INVALID INSTRUCTION FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an Invalid Instruction has been detected. Further detail will also be stored in the Invalid Instruction Extension byte (See Section 6.2.3).

A SNAP picture will be taken if enabled by the Snap Enable Indicator (Sec. 4.4.1) and the Snap Picture Enable (Sec. 4.6). See Section 6.2.3 for those errors classified as Instruction Errors.

6.2.1.6 (DIGIT 73 BIT 2) - UNCORRECTABLE MEMORY PARITY ERROR

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an Uncorrectable "multi-bit" Memory Parity Error has been detected. An Uncorrectable Memory Parity Error during the execution of a processor instruction will terminate the instruction without writing into memory at the location the error was detected.

6.2.1.7 (DIGIT 73 BIT 1) - ADDRESS ERROR FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an Address Error has been detected. Further detail will also be stored in the Address Error Extension byte (See Section 6.2.2).

A SNAP picture will be taken if enabled by the Snap Enable Indicator (Sec. 4.4.4) and the Snap Picture Enable (Sec. 4.6). See Section 6.2.2 for those errors classified as Address Errors.

6.2.1.8 (DIGIT 73 BIT 0) - INSTRUCTION TIMEOUT FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an instruction has taken longer than a specified processor dependent timeout value. The Instruction Timeout timer value varies from processor to processor. See Appendix A - Compatibility Notes (A.27.4).

A SNAP picture will be taken if enabled by the Snap Enable Indicator (Sec. 4.4.1) and the Snap Picture Enable (Sec. 4.6).

6.2.1.9 (DIGIT 74 BIT 3) - STACK OVERFLOW FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an attempted stack operation would have exceeded the limit of Memory Area "zero". This bit may only be set by the Virtual Enter (OP = 35), the Hyper Call (OP = 62) or the Adjust Stack Pointer (OP = 61) instructions.

6.2.1.10 (DIGIT 74 BIT 2) - ACCUMULATOR TRAP FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the failing instruction because an Accumulator Trap Fault occured.

6.2.1.11 (DIGIT 74 BIT 1) - SNAP PICTURE TAKEN

This fault indicates that a SNAP Picture was stored in memory at a location that has been previously been set with a Write Hardware Register instruction (OP = 65:BF = 01).

6.2.1.12 (DIGIT 74 BIT 0) - SOFT FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the next instruction to be executed because a Soft Fault has been detected. A Soft Fault is detected when Soft Fault Enable is set and the soft fault digit, located in the Reinstate List entry for the current task, is not equal to zero. This indicator may only be set by the Hyper Call (OP = 62), Branch Communicate (OP = 30), Return (OP = 63) and Virtual Branch Reinstate (OP = 93) instructions.

6.2.1.13 (DIGIT 75 BIT 0) - TASK TIMER FAULT

This fault indicates that a Hardware Call procedure was executed that stored the address of the next instruction to be executed because the task timer reached a value of zero. See Appendix A - Compatibility Notes (A.45).

6.2.2 (DIGIT 78-79) - ADDRESS ERROR EXTENSION (AEX)

When the Address Error Fault Indicator is set (See Section 6.2.1.7), The Address Error Extension byte will further define the type of error.

INFORMATION	VALUE
General	00
Invalid Address Relationship	01
Hyper Call Function Limit Error	02
Odd Operand Address	03
Invalid MAT Entry Type	04
Index Register, General	10
Invalid Arithmetic	11
Index Register Contains Undigit	12
Invalid Base Indicant	13
Stack Pointer (IX3) is negative	14
Stack Pointer (IX3) is odd	15
Base/Limit Error, General	20
Instruction Fetch	21
Address Resolution	22
Operand Write	23
Operand Read	24
Global Link Address	25
Address Undigit, General	30
Instruction Fetch	31
Address Resolution	32
Operand Write	33
Operand Read	34
Global Link Address	35
Branch Address, General	40
Address >= Limit	41
Address Contains Undigit	42
Odd Address	43

6.2.2 (DIGIT 78-79) - ADDRESS ERROR EXTENSION (Continued)

INFORMATION	VALUE
Invalid Environment Descriptor	50
Invalid Environment Number	51
Invalid Most Significant Digit	52
Index Contains Undigit	53
Memory Area Number Contains Undigit	54
Environment Number or Memory Area Number Out of Range, General	56
Environment Number Out of Range	57
Memory Area Number Out of Range	58
Invalid Memory Area Table Entry	60
Invalid Environment Number	61
Invalid Most Significant Digit	62
Index Contains Undigit	63
Memory Area Number Contains Undigit	64
Environment Number or Memory Area Number Out of Range, General	66
Environment Number Out of Range	67
Memory Area Number Out of Range	68
"E" Copy Type Entry points to a Copy Type Entry	69

6.2.3 (DIGIT 80-81) - INVALID INSTRUCTION EXTENSION (IEX)

When the Invalid Instruction Fault Indicator is set (See Section 6.2.1.5), the Invalid Instruction Extension byte will further define the type of error.

INFORMATION	VALUE
General	00
Invalid Operator Code	01
Privileged Mode Violation	02
Invalid Address Controller	03
Stack Overflow (OP = 31)	04
Counter Overflow	05
Invalid Field Comparison	06
Invalid Operand Field	07
Invalid AF or BF	20
Literal not Allowed	21
Invalid Literal	22
Invalid Indirect Field Length	23
Invalid Variant	24
Invalid AF Variant	25
Invalid BF Variant	26
Invalid Priviliged Primary Access	31
Invalid Priviliged Secondary Access	32
Invalid Attempt to Modify Original or Fault Memory Area Table Entry	35
Copy Protection Violation	36
Stack Protection Violation	37

7 INSTRUCTION SET SUMMARY

ADDR'S	=	Number of Instruction Addresses.
LG	=	Non-extended length of the instruction, in digits.
FLG	=	Comparison and Overflow flags changed.
AIN	=	AF indirect allowed.
BIN	=	BF indirect allowed.

SEC.	OP		NAME	ADDR'S	LG	LIT	AIN	BIN	FLG
8.	ARITHMETIC; Fixed point, Variable Field Length								
8.1	01	INC	Two-address Add	2	18	Y	Y	Y	Y
8.2	02	ADD	Three-address Add	3	24	Y	Y	Y	Y
8.3	03	DEC	Two-address Subtract	2	18	Y	Y	Y	Y
8.4	04	SUB	Three-address Subtract	3	24	Y	Y	Y	Y
8.5	05	MPY	Multiply	3	24	Y	Y	Y	Y
8.6	06	DIV	Divide	3	24	Y	Y	Y	Y
9.	ARITHMETIC; Fixed Point, Fixed Field Length								
9.1	58	ILD	Integer Load	1	8	N	N	N	Y
9.2	59	IST	Integer Store	1	8	N	N	N	Y
9.3	50	IAD	Integer Add	1	8	N	N	N	Y
9.4	51	IAS	Integer Add/Store	1	8	N	N	N	Y
9.5	52	ISU	Integer Subtract	1	8	N	N	N	Y
9.6	53	ISS	Integer Subtract/Store	1	8	N	N	N	Y
9.7	54	IMU	Integer Multiply	1	8	N	N	N	Y
9.8	55	IMS	Integer Multiply/Store	1	8	N	N	N	Y
9.9	57	IMI	Increment Memory	1	8	N	N	N	Y
10.	ARITHMETIC; Floating Point, Fixed Field Length								
10.1	78	RLD	Real Load	1	8	N	N	N	Y
10.2	79	RST	Real Store	1	8	N	N	N	Y
10.3	70	RAA	Real Add	1	8	N	N	N	Y
10.4	71	RAS	Real Add/Store	1	8	N	N	N	Y
10.5	72	RSU	Real Subtract	1	8	N	N	N	Y
10.6	73	RSS	Real Subtract/Store	1	8	N	N	N	Y
10.7	74	RMU	Real Multiply	1	8	N	N	N	Y
10.8	75	RMS	Real Multiply/Store	1	8	N	N	N	Y
10.9	76	RDV	Real Divide	1	8	N	N	N	Y
10.10	77	RDS	Real Divide/Store	1	8	N	N	N	Y
10.11	84	ACM	Accumulator Manipulate	0	4	N	N	N	Y

7 INSTRUCTION SET SUMMARY (Continued)

SEC.	OP		NAME	ADDR'S	LG	LIT	AIN	BIN	FLG
11.	ADDRESS BRANCHING								
11.1	20	NOP	No Operation	1	8	N	N	N	N
11.1	21	LSS	Branch on Less Than	1	8	N	N	N	N
11.1	22	EQL	Branch on Equal	1	8	N	N	N	N
11.1	23	LEQ	Branch on Less Than or Equal	1	8	N	N	N	N
11.1	24	GTR	Branch on Greater Than	1	8	N	N	N	N
11.1	25	NEQ	Branch on Not Equal	1	8	N	N	N	N
11.1	26	GEQ	Branch on Greater Than or Equal	1	8	N	N	N	N
11.1	27	BUN	Branch Unconditional	1	8	N	N	N	N
11.1	28	OFL	Branch on Overflow	1	8	N	N	N	N
11.1	2A	NUL	Branch on Null	1	8	N	N	N	N
11.1	2B	GTN	Branch on Greater or Null	1	8	N	N	N	N
11.1	B1	LSS	Branch on Less Than	1	8	N	N	N	N
11.1	B2	EQL	Branch on Equal	1	8	N	N	N	N
11.1	B3	LEQ	Branch on Less Than or Equal	1	8	N	N	N	N
11.1	B4	GTR	Branch on Greater Than	1	8	N	N	N	N
11.1	B5	NEQ	Branch on Not Equal	1	8	N	N	N	N
11.1	B6	GEQ	Branch on Greater Than or Equal	1	8	N	N	N	N
11.1	BA	NUL	Branch on Null	1	8	N	N	N	N
11.1	BB	GTN	Branch on Greater or Null	1	8	N	N	N	N
11.1	E1	LSS	Branch on Less Than	1	8	N	N	N	N
11.1	E2	EQL	Branch on Equal	1	8	N	N	N	N
11.1	E3	LEQ	Branch on Less Than or Equal	1	8	N	N	N	N
11.1	E4	GTR	Branch on Greater Than	1	8	N	N	N	N
11.1	E5	NEQ	Branch on Not Equal	1	8	N	N	N	N
11.1	E6	GEQ	Branch on Greater Than or Equal	1	8	N	N	N	N
11.1	EA	NUL	Branch on Null	1	8	N	N	N	N
11.1	EB	GTN	Branch on Greater or Null	1	8	N	N	N	N

7 INSTRUCTION SET SUMMARY (Continued)

SEC.	OP		NAME	ADDR'S	LG	LIT	AIN	BIN	FLG
11.	ADDRESS BRANCHING								
11.1	F1	LSS	Branch on Less Than	1	8	N	N	N	N
11.1	F2	EQL	Branch on Equal	1	8	N	N	N	N
11.1	F3	LEQ	Branch on Less Than or Equal	1	8	N	N	N	N
11.1	F4	GTR	Branch on Greater Than	1	8	N	N	N	N
11.1	F5	NEQ	Branch on Not Equal	1	8	N	N	N	N
11.1	F6	GEQ	Branch on Greater Than or Equal	1	8	N	N	N	N
11.1	FA	NUL	Branch on Null	1	8	N	N	N	N
11.1	FB	GTN	Branch on Greater or Null	1	8	N	N	N	N
12.	HALTS								
12.1	29	HBR	Halt Branch	1	8	N	N	N	N
12.2	48	HBK	Halt Breakpoint	0	6	N	Y	N	N
13.	ENVIRONMENT CHANGE								
13.1	30	BCT	Branch Communicate	0	6	N	Y	Y	Y
13.2	31	NTR	Enter	1	12+	N	Y	Y	Y
13.3	32	EXT	Exit	1	8	N	N	N	Y
13.4	35	VEN	Virtual Enter	2	18	Y+	Y	Y	Y
13.5	62	HCL	Hyper Call	2	18	Y+	Y	Y	Y
13.6	63	RET	Return	0	4	N	Y	Y	Y
13.7	61	ASP	Adjust Stack Pointer	1	12	Y	Y	Y	N
13.8	90	INT	Interrupt	1	12	N	Y	Y	N
13.9	93	BRV	Virtual Branch Reinstate	0	4	N	Y	N	Y

+ = n Characters (Stack Parameters) follow as specified by the instruction (0=<n=<9,999 bytes).

Y+ = Special literal value.

7 INSTRUCTION SET SUMMARY (Continued)

SEC.	OP		NAME	ADDR'S	LG	LIT	AIN	BIN	FLG
14.	DATA MOVEMENT								
14.1	08	MVD	Move Data	3	24	N	Y	Y	N
14.2	09	MVL	Move Links	3	24	N	Y	Y	N
14.3	10	MVA	Move Alphanumeric	2	18	Y	Y	Y	Y
14.4	11	MVN	Move Numeric	2	18	Y	Y	Y	Y
14.5	12	MVW	Move Words	2	18	N	Y	Y	N
14.6	13	MVC	Move and Clear Words	2	18	N	Y	Y	N
14.7	14	MVR	Move Repeat	2	18	Y	Y	Y	N
14.8	15	TRN	Translate	3	24	N	Y	Y	N
14.9	49	EDT	Edit	3	24	Y	Y	Y	Y
15.	LOGICAL								
15.1	16	SDE	Scan-Delimiter Equal	2	18	Y	Y	Y	Y
15.2	17	SDU	Scan-Delimiter Unequal	2	18	Y	Y	Y	Y
15.3	18	SZE	Scan-Zone Equal	2	18	Y	Y	Y	Y
15.4	19	SZU	Scan-Zone Unequal	2	18	Y	Y	Y	Y
15.5	39	SEA	Search	3	24	Y	Y	Y	Y
15.6	37	SLL	Search Link List	2	18	Y	Y	Y	Y
15.7	38	SLD	Search Link Delink	2	18	Y	Y	Y	Y
15.8	64	SLT	Search List	3	24	Y	Y	Y	Y
15.9	66	STB	Search Table	3	24	Y	Y	Y	Y
15.10	40	BZT	Bit Zero Test	1	12	Y	Y	N	Y
15.11	41	BOT	Bit One Test	1	12	Y	Y	N	Y
15.12	45	CPA	Compare Alphanumeric	2	18	Y	Y	Y	Y
15.13	46	CPN	Compare Numeric	2	18	Y	Y	Y	Y
15.14	33	BRT	Bit Reset	1	12	N	Y	N	Y
15.15	34	BST	Bit Set	1	12	N	Y	N	Y
15.16	42	AND	Logical And	3	24	Y	Y	Y	Y
15.17	43	ORR	Logical Or	3	24	Y	Y	Y	Y
15.18	44	NOT	Logical Not	3	24	Y	Y	Y	Y

7 INSTRUCTION SET SUMMARY (Continued)

SEC.	OP		NAME	ADDR'S	LG	LIT	AIN	BIN	FLG
16.	INPUT/OUTPUT								
16.1	94	IIO	Initiate I/O	1	12	Y	Y	Y	Y
16.2	92	RAD	Read address	1	12	N	Y	Y	Y
16.3	91	SRD	Scan Result Descriptor	0	6	N	Y	Y	Y
16.4	85	CIO	Convert I/O	2	18	N	Y	Y	Y
16.5	98	IOC	I/O Complete	2	18	Y	Y	Y	Y
17.	BINARY/DECIMAL CONVERSION								
17.1	88	D2B	Decimal to Binary	2	18	Y	Y	Y	Y
17.2	89	B2D	Binary to Decimal	2	18	Y	Y	Y	Y
18.	TIME-OF-DAY TIMER								
18.1	95	RDT	Read Time of Day	1	12	N	Y	Y	N
18.2	97	STT	Set Time of Day	1	12	N	Y	Y	N
19.	MEASUREMENT								
19.1	87	MOP	Measurement OP	2	18	N	Y	Y	N
20.	MISCELANEOUS								
20.1	86	ATE	Alter Table Entry	2	18	N	Y	Y	N
20.2	67	LIX	Load Index Registers	1	12	N	Y	Y	N
20.3	68	SIX	Store Index Registers	1	12	N	Y	Y	N
20.4	60	LOK	Lock/Unlock	1	12	N	Y	Y	Y
20.7	65	WHR	Write Hardware Registers	1	12	N	Y	Y	N
20.8	47	SMF	Set Mode	0	6	N	Y	Y	N
20.9	AB	BAD	Fail	1	12	N	N	N	N
20.10	99	SST	System Status	1	12	N	Y	Y	N
21.	STRING								
21.1	A0	MVS	Move Strings	2	18	N	Y	Y	Y
21.2	A1	CPS	Compare Strings	2	18	N	Y	Y	Y
21.3	A2	HSH	Hash Strings	2	18	N	Y	Y	N

8 ARITHMETIC; FIXED POINT, VARIABLE FIELD LENGTH

8.1 TWO ADDRESS ADD (INC)/OP=01

Format

```
+----+----+----+-----------+-----------+
| OP | AF | BF |     A     |     B     |
+----+----+----+-----------+-----------+
```

OP = 01

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the addend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the augend and sum field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

Function

The Two Address Add instruction adds the contents of one memory location (A) to the contents of a second memory location (B) and stores the sum in the second memory location (B) unless an overflow condition exists. If the number of significant digits in the result is greater than the sum field length, the sum field will be unchanged, the Comparison Flags will be unchanged and the Overflow Flag will be set.

8.1 TWO-ADDRESS ADD (INC)/OP=01 (Continued)

Store the absolute value of the sum when the sum field data type is unsigned (UN or UA). Store the standard EBCDIC form of the result sign as the first digit of the result when the sum field data type is SN. Fill the zone digit with the EBCDIC numeric subset code (F) when the sum field data type is alphanumeric (UA).

The sign of a zero is always considered to be positive.

If the addend and the augend are of unequal lengths (AF not equal to BF), the shorter of the two is treated as if it has been left filled with zero's.

Only the numeric digits of an alphanumeric field enter into the operation. Unsigned (UN or UA) operands are assumed to be positive.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison Flags

In all cases except overflow, set the Comparison Flags to indicate whether the sum is greater than (HIGH), equal to (EQUAL), or less than (LOW) zero.

Overlap

"A" and "B" may totally overlap or may have matching type-address overlap (See 4.9.4). Partial overlap of "A" and "B" other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.08).

8.1 TWO-ADDRESS ADD (INC)/OP=01 (Continued)

Examples

EXAMPLE (1) Add an Alpha Field to a Signed Field

```
OP AF BF       A             B
-- -- --      ---           ---
01 02 04, A FIELD (UA), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	C1E7	unchanged
B FIELD	+0257	C0274
COMPARISON	nn	HIGH
OVERFLOW	nn	unchanged

EXAMPLE (2) Add with Overflow condition

```
OP AF BF       A             B
-- -- --      ---           ---
01 02 03, A FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	18	unchanged
B FIELD	985	unchanged
COMPARISON	nn	unchanged
OVERFLOW	nn	ON

8.2 THREE ADDRESS ADD (ADD)/OP=02

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 02

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the addend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the augend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

C = Address of the sum field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

Function

The Three Address Add instruction adds the contents of one memory location (A) to the contents of a second memory location (B) and stores the sum in a third memory location (C) unless an overflow condition exists. The sum field length is equal to the larger of the AF and BF values. If the number of significant digits in the result is greater than the sum field length, the sum field will be unchanged, the Comparison Flags will be unchanged and the Overflow Flag will be set.

8.2 THREE ADDRESS ADD (ADD)/OP=02 (Continued)

Store the absolute value of the sum when the sum field data type is unsigned (UN or UA). Store the standard EBCDIC form of the result sign as the first digit of the result when the sum field data type is SN. Fill the zone digit with the EBCDIC subset code (F) when the sum field data type is alphanumeric (UA).

The sign of a zero sum is always positive.

If the addend and augend are of unequal lengths, (AF not equal to BF), the shorter of the two will be treated as if it has been left filled with zero's.

Only the numeric digits of an alphanumeric field enter into the operation.

Unsigned (UN or UA) operands are assumed to be positive.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison Flags

In all cases except overflow, set the Comparison Flags to indicate whether the sum is greater than (HIGH), equal to (EQUAL), or less than (LOW) zero.

Overlap

"A" and "B" may partially or totally overlap. "A" or "B" may totally overlap with "C", or may have matching type-address overlap (see 4.9.4).

Partial overlap of "A" or "B" with "C", other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.09).

8.2 THREE ADDRESS ADD (ADD)/OP=02 (Continued)

Examples

EXAMPLE (1) Add an Unsigned Field to a Signed Field giving a Signed Sum

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
02 02 05, A FIELD (UN), B FIELD (SN), C FIELD (SN)

                        BEFORE      AFTER

  A FIELD                   20    unchanged
  B FIELD               +00015    unchanged
  C FIELD               nnnnnn      C00035

  COMPARISON             nnn         HIGH
  OVERFLOW               nnn      unchanged
```

EXAMPLE (2) Add an Unsigned Field to a Signed Field giving an Unsigned Alpha Sum

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
02 02 05, A FIELD (UN), B FIELD (SN), C FIELD (UA)

                        BEFORE      AFTER

  A FIELD                   10    unchanged
  B FIELD               D00050    unchanged
  C FIELD           nnnnnnnnnn   F0F0F0F4F0

  COMPARISON             nnn          LOW
  OVERFLOW               nnn      unchanged
```

8.2 THREE ADDRESS ADD (ADD)/OP=02 (Continued)

EXAMPLE (3) Add an Alpha Field to an Alpha Field

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
02 02 02, A FIELD (UA), B FIELD (UA), C FIELD (UA)
```

	BEFORE	AFTER
A FIELD	F4F0	unchanged
B FIELD	C1C2	unchanged
C FIELD	nnnn	F5F2
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (4) Add two fields with an Overflow Condition.

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
02 02 02, A FIELD (UN), B FIELD (UN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	61	unchanged
B FIELD	53	unchanged
C FIELD	nn	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

8.3 TWO ADDRESS SUBTRACT (DEC)/OP=03

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 03

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length 100 units.

A = Address of the subtrahend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the minuend/difference field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

Function

The Two Address Subtract instruction subtracts the contents of memory location (A) from the contents of a second memory location (B) and stores the difference in the second memory location (B) unless an overflow condition exists.

If the number of significant digits in the result is greater than the difference field length, the difference field will be unchanged, the Comparison Flags will be unchanged and the Overflow Flag will be set.

Store the absolute value of the difference when the difference field is unsigned (UN or UA). Store the standard EBCDIC form of the result sign as the first digit of the result when the difference field data type is SN. fill the zone digit with the EBCDIC numeric subset code (F) when the difference field data type is alphanumeric (UA).

8.3 TWO ADDRESS SUBTRACT (DEC)/OP=03 (Continued)

The sign of a zero difference is always positive.

If the subtrahend and minuend are of unequal length (AF not equal to BF), the shorter of the two is treated as if it has been left filled with zero's.

Only the numeric digits of an alphanumeric field enter into the operation. Unsigned (UN or UA) operands are assumed to be positive.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison Flags

In all cases, except overflow, set the Comparison Flags to indicate whether the difference is greater than (HIGH), equal to (EQUAL), or less than (LOW) zero.

Overlap

"A" and "B" may totally overlap or may have matching type-address overlap (See 4.9.4). Partial overlap of "A" and "B" other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.08).

8.3 TWO ADDRESS SUBTRACT (DEC)/OP=03 (Continued)

Examples

EXAMPLE (1) Subtract two positive numbers

```
OP AF BF       A            B
-- -- --      ---          ---
03 03 03, A FIELD (SN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	+014	unchanged
B FIELD	+062	C048
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (2) Subtract two Negative numbers

```
OP AF BF       A            B
-- -- --      ---          ---
03 03 03, A FIELD (SN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D035	unchanged
B FIELD	D029	C006
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

8.3 TWO ADDRESS SUBTRACT (DEC)/OP=03 (Continued)

EXAMPLE (3) Subtract Signed Field From Unsigned Field

```
OP AF BF      A           B
-- -- --     ---         ---
03 02 03, A FIELD (SN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	D71	unchanged
B FIELD	121	192
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (4) Subtract Unsigned Field from Signed Field

```
OP AF BF      A            B
-- -- --     ---          ---
03 03 03, A FIELD, (UN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	259	unchanged
B FIELD	+138	D121
COMPARISON	nnn	LOW
OVERFLOW	nnn	unchanged

8.3 TWO ADDRESS SUBTRACT (DEC)/OP=03 (Continued)

EXAMPLE (5) Subtract Two Signed Fields, Overflow Condition

```
OP AF BF      A            B
-- -- --     ---          ---
03 03 03, A FIELD (SN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D556	unchanged
B FIELD	+942	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

8.4 THREE ADDRESS SUBTRACT (SUB)/OP=04

<u>Format</u>

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 04

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the subtrahend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the minuend field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

C = Address of the difference field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

<u>Function</u>

The Three Address Subtract instruction subtracts the contents of one memory location (A) from the contents of a second memory location (B) and stores the difference in a third memory location (C) unless an overflow condition exists. The difference field length is equal to the larger of AF or BF.

If the number of significant digits in the result is greater than the difference field length, the difference field will be unchanged, the Comparison Flags will be unchanged and the Overflow Flag will be set.

8.4 THREE ADDRESS SUBTRACT (SUB)/OP=04 (Continued)

Store the absolute value of the difference when the difference field is unsigned (UN or UA). Store the standard EBCDIC form of the result sign as the first digit of the result when the difference field data type is SN. Fill the zone digit with the EBCDIC numeric subset code (F) when the difference field data type is alphanumeric (UA).

The sign of a zero difference is always positive.

If the subtrahend and minuend are of unequal length (AF not equal to BF), the shorter field is treated as if it has been left filled with zero's.

Only the numeric digits of an alphanumeric field enter into the operation.

Unsigned (UN or UA) fields are assumed to be positive.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison Flags

In all cases, except overflow, set the Comparison Flags to indicate whether the difference is greater than (HIGH), equal to (EQUAL), or less than (LOW) zero.

Overlap

"A" and "B" may partially or totally overlap. "A" or "B" may totally overlap with "C", or may have matching type-address overlap. (See 4.9.4)

Partial overlap of "A" or "B" with "C", other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.09).

8.4 THREE ADDRESS SUBTRACT (SUB)/OP=04 (Continued)

Examples

EXAMPLE (1) Subtract an Unsigned Field from an Alpha Field

```
OP AF BF        A              B              C
-- -- --       ---            ---            ---
04 01 05, A FIELD (UN), B FIELD (UA), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	5	unchanged
B FIELD	C1C2C3C4C5	unchanged
C FIELD	nnnnnn	C12340
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

8.5 MULTIPLY (MPY)/OP=05

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 05

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of"00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the multiplier field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the multiplicand field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

C = Address of the product field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

Function

The multiply instruction multiplies the contents of one memory location (B) by the contents of a second memory location (A) and stores the product in a third memory location (C). The product field length is the sum of AF and BF, and could be as long as 200 units.

Store the absolute value of the product when the product field data type is unsigned (UN or UA). Store the standard EBCDIC form of the sign as the first digit of the result when the product field data type is SN. Fill the zone digit with the EBCDIC numeric subset code (F) when the product field data type is alphanumeric (UA).

8.5 MULTIPLY (MPY)/OP=05 (Continued)

The Overflow Flag is not affected by this instruction.

The sign of a zero product is always positive.

Only the numeric digits of an alphanumeric field (UA) enter into the operation. Unsigned fields are assumed to be positive.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison Flags

In all cases, set the Comparison Flags to indicate whether the product is positive (HIGH), equal to zero (EQUAL) or negative (LOW).

Overlap

"A" and "B" may partially or totally overlap.

Overlap of "A" and "B" with "C", other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.10).

8.5 MULTIPLY (MPY)/OP=05 (Continued)

Examples

EXAMPLE (1) Multiply an Alpha Field by an Unsigned Field

```
OP AF BF        A              B              C
-- -- --       ---            ---            ---
05 02 05, A FIELD (UA), B FIELD (UN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D1D2	unchanged
B FIELD	00011	unchanged
C FIELD	nnnnnnnn	C0000132
COMPARISON	nnn	HIGH

EXAMPLE (2) Multiply Two Signed Numbers

```
OP AF BF        A              B              C
-- -- --       ---            ---            ---
05 02 02, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D15	unchanged
B FIELD	D17	unchanged
C FIELD	nnnnn	C0255
COMPARISON	nnn	HIGH

8.6 DIVIDE (DIV)/OP=06

Format

```
+----+----+----+--------+--------+--------+
| OP | AF | BF |   A    |   B    |   C    |
+----+----+----+--------+--------+--------+
```

OP = 06

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the divisor field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

B = Address of the dividend/remainder field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

C = Address of the quotient field. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

Function

The divide instruction divides the contents of one memory location (B) by the contents of a second memory location (A) storing the remainder in the "B" data field and storing the quotient in a third memory location (C).

The length of the dividend field must be greater than the length of the divisor field (BF greater than AF). The length of the quotient field is the difference in length of the "A" and "B" fields (BF-AF). If the result is too large to fit into the quotient field or if BF is not greater than AF, the division is not performed, the contents of "B" and "C" are unchanged, the Comparison Flags are unchanged, and the Overflow Flag is set.

8.6 DIVIDE (DIV)/OP=06 (Continued)

If the absolute value of the divisor is not greater than the absolute value of the equivalent number of leading digits of the dividend, the division is not performed and the Overflow Flag is set with the Comparison Flags remaining unchanged. Note that a divisor which is zero will fail this test and the Overflow Flag will be set.

The absolute value of the quotient is stored when the quotient field data type is unsigned (UN or UA). The standard EBCDIC form of the sign is stored as the first digit of the result when the quotient data type is SN. The zone digits are filled with the EBCDIC numeric subset code (F) when the quotient field data type is alphanumeric (UA).

The absolute value of the remainder is stored when the remainder field data type is unsigned (UN or UA). The standard EBCDIC form of the sign is stored as the first digit of the result when the remainder data type is SN. The zone digits are filled with the EBCDIC numeric subset code (F) when the remainder field data type is alphanumeric (UA).

Only the numeric digits of an alphanumeric field (UA) enter into the operation.

Unsigned fields are assumed to be positive.

The sign of the quotient is positive if the sign of the divisor and the dividend are the same or the quotient is zero, otherwise the sign is negative.

If the dividend data type is SN, the sign of the dividend will be left unchanged in memory and will thus become the sign of the remainder. Therefore this final remainder sign could be other than "C" or "D" and a remainder of zero magnitude could have a negative sign.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

8.6 DIVIDE (DIV)/OP=06 (Continued)

Comparison Flags

In all cases, except overflow, set the Comparison Flags to indicate whether the quotient is positive (HIGH), equal to zero (EQUAL) or negative (LOW).

Overlap

Partial overlap of the dividend field (B) and either of the other operands may produce inconsistent results.

If the address of the dividend field is the same as the address of the quotient field (B = C) and the respective address controllers are equal (BC = CC), a result will be produced that consists of the quotient followed by the least significant AF units of the remainder. In the case of SN data, the sign of the quotient will be stored in the first digit of the result followed by the quotient and the least significant AF digits of the remainder.

Examples

EXAMPLE (1) Divide Two Signed Numbers

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
06 01 04, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER	
A FIELD	+9	unchanged	
B FIELD	+0101	+0002	Remainder
C FIELD	nnnn	C011	Quotient
COMPARISON	nnn	HIGH	
OVERFLOW	nnn	unchanged	

8.6 DIVIDE (DIV)/OP=06 (Continued)

EXAMPLE (2) Divide Two Signed Fields, Negative Numbers

```
OP AF BF       A            B             C
-- -- --      ---          ---           ---
06 02 05, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER	
A FIELD	D12	unchanged	
B FIELD	D00187	D00007	Remainder
C FIELD	nnnn	C015	Quotient
COMPARISON	nnn	HIGH	
OVERFLOW	nnn	unchanged	

EXAMPLE (3) Divide Producing Overflow (Length Problem)

```
OP AF BF       A            B             C
-- -- --      ---          ---           ---
06 04 03, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D1014	unchanged
B FIELD	D123	unchanged
C FIELD	nnnn	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

EXAMPLE (4) Divide Producing Overflow (Data Problem)

```
OP AF BF       A            B             C
-- -- --      ---          ---           ---
06 02 03, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D11	unchanged
B FIELD	D125	unchanged
C FIELD	nnnn	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

8.6 DIVIDE (DIV)/OP=06 (Continued)

EXAMPLE (5) Divide By Zero

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
06 02 03, A FIELD (SN), B FIELD (SN), C FIELD (SN)
```

	BEFORE	AFTER
A FIELD	+00	unchanged
B FIELD	D125	unchanged
C FIELD	nnnn	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

EXAMPLE (6) Total Overlap of "B" & "C".

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
06 02 04, A FIELD (UN), B FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	13	unchanged
B FIELD	1127	8609
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9 ARITHMETIC; FIXED POINT, FIXED FIELD LENGTH

Fixed field length arithmetic instructions use a 20 digit accumulator which holds the instruction result within the processor as an operand for a subsequent operation. Every instruction has an implied reference to the accumulator.

The fixed point (integer) format consists of an implied signed exponent field (+08) (of 3 digits) followed by a mantissa field of a single sign digit followed by eight digits of mantissa.

The fixed point instructions operate on the twelve most significant digits of the accumulator. This format is similar to the single Precision format in the Floating Point, Fixed field length instructions (See Section 10.). The same accumulator is used for the Fixed Point and Floating Point instructions.

If the instruction produces a result greater than seven digits, an overflow occurs and the following characteristics apply.

1. If the operation specifies a store of the result to memory (IAS, ISS, IMS, IMI), this store is not performed.

2. The sign and exponent field is set to +08.

3. Set the Overflow Flag and set the Comparison Flags to HIGH.

4. The final contents of the accumulator are unspecified.

9 ARITHMETIC; FIXED POINT, FIXED FIELD LENGTH (Continued)

A Trap Fault is a software enabled routine that allows the instruction in error to be examined.

The Trap Fault will be enabled if the two digit key stored at memory address 64, relative to Base #0, is equal to "FF".

If Trap Fault is enabled and a fault occurs, a Hardware Call procedure will be executed with the address of the instruction at fault stored on the stack.

If Trap Fault is not enabled and a fault occurs, the next program instruction will be executed.

9.1 INTEGER LOAD (ILD)/OP=58

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 58

A = Address of the source data field. Address may be indexed, indirect or extended. The final address controller data type will always be treated as SN.

Function

The integer load instruction loads the accumulator with an 8 digit data field at memory (A). The data field, consisting of a sign and seven digits, is loaded into the 20 digit accumulator as shown below, where "S" is the sign of the data field and "Dn" represents the numeric data. Undigits may be loaded into the mantissa field of the accumulator. The result of loading undigits into the sign digit is machine dependent. See Appendix A - Compatibility Notes (A.54).

```
+---------+-----------+-----------+---------+---------+
| + 0 8 S | 0 D2 D3 D4 | D5 D6 D7 D8 | 0 0 0 0 | 0 0 0 0 |
+---------+-----------+-----------+---------+---------+
```

Note that the exponent is set to +08 and that the most siginificent digit as well as the least significant eight digits of the mantissa are set to zero.

Comparison Flags

Set the Comparison Flags HIGH if the result is positive, EQUAL if the result equal to zero, and LOW if the result is negative.

Overflow

The Overflow Flag is not affected by this instruction.

9.1 INTEGER LOAD (ILD)/OP=58 (Continued)

Examples

EXAMPLE (1) Load Accumulator with Integer

```
OP       A
--      ---
58    A FIELD

                      BEFORE            AFTER

A FIELD             D9876543          unchanged
ACCUMULATOR       nnnnnnnnnnnn      +08-09876543
                      nnnnnnnn          00000000

COMPARISON            nnn                LOW
OVERFLOW              nnn             unchanged
```

EXAMPLE (2) Load Accumulator with Undigits

```
OP       A
--      ---
58    A FIELD

                      BEFORE            AFTER

A FIELD             D0F1B2E3          unchanged
ACCUMULATOR       nnnnnnnnnnnn      +08-00F1B2E3
                      nnnnnnnn          00000000

COMPARISON            nnn                LOW
OVERFLOW              nnn             unchanged
```

9.2 INTEGER STORE (IST)/OP=59

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 59

A = Address of the destination field. Address may be indexed, indirect or extended. The final address controller data type will always be treated as SN.

Function

The integer store instruction will store into an 8 digit field at memory location (A) the integer and its sign from the accumulator. Undigits may be stored from the mantissa field of the accumulator. The handling of the sign digit is machine dependent. See Appendix A - Compatibility Notes (A.54).

The accumulator, which is in the form:

```
+---------+-------------+-------------+-------------------------+
| + 0 8 S |D1 D2 D3 D4  | D5 D6 D7 D8 |D9D10D11D12D13D14D15D16|
+---------+-------------+-------------+-------------------------+
```

is stored in the destination field as:

```
+------------+------------+
| S D2 D3 D4 | D5 D6 D7 D8 |
+------------+------------+
```

Where "S" indicates the operand sign and "Dn" represents the operand digits. Notice that D1 is dropped. Meaningful results are obtained only when D1 is equal to zero.

Comparison Flags
---------- -----

Set the Comparison Flags HIGH if the stored operand is positive, EQUAL if the stored operand is equal to zero, and LOW if the stored operand is negative.

9.2 INTEGER STORE (IST)/OP=59 (Continued)

Overflow

The Overflow Flag is not affected by this instruction.

Examples

EXAMPLE (1) Store Accumulator Integer in Memory

OP	A
59	A FIELD

	BEFORE	AFTER
A FIELD	nnnnnnnn	D9876543
ACCUMULATOR	+08-09876543 nnnnnnnn	unchanged
COMPARISON	nnn	LOW

EXAMPLE (2) Store Accumulator Image in Memory

OP	A
59	A FIELD

	BEFORE	AFTER
A FIELD	nnnnnnnn	D0F1B2E3
ACCUMULATOR	+08-00F1B2E3 nnnnnnnn	unchanged
COMPARISON	nnn	LOW

9.3 INTEGER ADD (IAD)/OP=50

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 50

A = Address of the Addend field. Address may be indexed, indirect or extended. The data type of the final address controller is ignored and will always be treated as SN.

Function

The integer add instruction adds the number stored in a memory location (A) to the value stored in the accumulator and stores the sum in the accumulator.

The signs of both the accumulator and the addend are considered in the addition and the mantissa sign field is set positive or negative based on the result.

Set the signed exponent field of the accumulator to +08 even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison and Overflow Flags

If the addition produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, the Comparison Flags will be set to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

9.3 INTEGER ADD (IAD)/OP=50 (Continued)

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Add Integer to Accumulator

```
OP      A
--     ---
50   A FIELD
```

	BEFORE	AFTER
A FIELD	+1111111	unchanged
ACCUMULATOR	+08+01234567	+08+02345678
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.4 INTEGER ADD AND STORE (IAS)/OP=51

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 51

A = Address of the Addend and Sum field. Address may be indexed, indirect or extended. The data type of the final address controller will always be treated as SN.

Function

The integer add and store instruction will add the number stored in a memory location (A) to the value stored in the accumulator and store the sum in the accumulator and in the same memory location (A). The store to memory does not take place on overflow.

The signs of both the accumulator and the addend are considered in the addition and the mantissa sign field is set positive or negative based on the result.

Set the signed exponent field of the accumulator to +08 even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

9.4 INTEGER ADD AND STORE (IAS)/OP=51 (Continued)

Comparison and Overflow Flags

If the addition produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. The Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, the Comparison Flags will be set to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Add Integer to Accumulator and Store

OP	A
51	A FIELD

	BEFORE	AFTER
A FIELD	+1111111	C2345678
ACCUMULATOR	+08+01234567	+08+02345678
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.5 INTEGER SUBTRACT (ISU)/OP=52

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 52

A = Address of the subtrahend field. Address may be indexed, indirect or extended. The data type of the final address controller will always be treated as SN.

Function

The integer subtract instruction will subtract the number stored in a memory location (A) from the value stored in the accumulator and store the difference in the accumulator.

The signs of both the accumulator and the subtrahend are considered in the subtraction and the mantissa sign field is set positive or negative based on the result.

The signed exponent field of the accumulator is set to +08 by this instruction even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison and Overflow Flags

If the subtraction produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, set the Comparison Flags to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

9.5 INTEGER SUBTRACT (ISU)/OP=52 (Continued)

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Subtract Integer from Accumulator

```
OP      A
--     ---
52  A FIELD
```

	BEFORE	AFTER
A FIELD	D0999999	unchanged
ACCUMULATOR	+08+02345678	+08+03345677
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.6 INTEGER SUBTRACT AND STORE (ISS)/OP=53

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 53

A = Address of the subtrahend and the difference field. Address may be indexed, indirect or extended. The data type of the final address controller will always be treated as SN.

Function

The integer subtract instruction will subtract the number stored in a memory location (A) from the value stored in the accumulator and store the difference in the accumulator and in the same memory location (A). The store to memory does not take place on overflow.

The signs of both the accumulator and the subtrahend are considered in the subtraction and the mantissa sign field is set positive or negative based on the result.

Set the signed exponent field of the accumulator to +08 even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

9.6 INTEGER SUBTRACT AND STORE (ISS)/OP=53 (Continued)

Comparison and Overflow Flags

If the subtraction produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, set the Comparison Flags to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Subtract Integer from Accumulator and Store

```
OP     A
--    ---
53  A FIELD
```

	BEFORE	AFTER
A FIELD	D0999999	C3345677
ACCUMULATOR	+08+02345678	+08+03345677
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.7 INTEGER MULTIPLY (IMU)/54

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 54

A = Address of the multiplier field. Address may be indexed, indirect or extended. The data type of the final address controller will always be treated as SN.

Function

The integer multiply instruction causes the value stored in the accumulator to be multiplied by the number stored in a memory location (A) and the product to be stored in the accumulator.

The signs of both the accumulator and the multiplicand are considered in the multiplication and the mantissa sign field is set positive or negative based on the result.

Set the signed exponent field of the accumulator to +08 even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Note, Section A.16.

Comparison and Overflow Flags
---------- --- -------- -----

If the multiplication produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, set the Comparison Flags to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

9.7 INTEGER MULTIPLY (IMU)/54 (Continued)

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Multiply Accumulator by Integer

OP	A
54	A FIELD

	BEFORE	AFTER
A FIELD	+0000003	unchanged
ACCUMULATOR	+08+01234567	+08+03703701
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.8 INTEGER MULTIPLY AND STORE (IMS)/OP=55

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 55

A = Address of the multiplier and the product field. Address may be indexed, indirect or extended. The data type of the final address controller will always be treated as SN.

Function

The integer multiply and store instruction causes the value stored in the accumulator to be multiplied by the number stored in a memory location (A), the product to be stored in the accumulator and in the same the memory location (A). The store to memory does not take place on overflow.

The signs of both the accumulator and the multiplicand are considered in the multiplication and the mantissa sign field is set positive or negative based on the result.

The signed exponent field of the accumulator is set to +08 by this instruction even in the case of overflow.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

9.8 INTEGER MULTIPLY AND STORE (IMS)/OP=55 (Continued)

Comparison and Overflow Flags

If the multiplication produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, set the Comparison Flags to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Multiply Accumulator by Integer and Store

OP	A
55	A FIELD

	BEFORE	AFTER
A FIELD	+0000003	C3703701
ACCUMULATOR	+08+01234567	+08+03703701
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

9.9 INTEGER MEMORY INCREMENT (IMI)/OP=57

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 57

A = Address of the increment field. Address may be indexed, indirect or extended. The final address controller if equal to one (SN) indicates a decrement operation and if equal to zero (UN) indicates an increment operation. Other controller values are reserved.

Function

The integer memory increment instruction, depending on the value of the address controller, increments or decrements a number at a memory location (A) and stores that value in the accumulator and at the same memory location (A).

The signed exponent field of the accumulator is always set to +08 by this instruction.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

Comparison and Overflow Flags

If the increment/decrement produces a result greater than 7 digits, set the Overflow Flag and set the Comparison Flags to HIGH. A Trap Fault, if enabled, will cause a Hardware Call procedure to the fault routine. The final contents of the accumulator are unspecified. If there is no overflow condition, set the Comparison Flags to EQUAL if the result is zero, HIGH if the result is positive and LOW if the result is negative.

9.9 INTEGER MEMORY INCREMENT (IMI)/OP=57 (Continued)

Overlap

There are no overlap restrictions for this instruction.

Examples

EXAMPLE (1) Memory Increment

```
OP      A
--     ---
57  A FIELD (UN)
```

	BEFORE	AFTER
A FIELD	+1234567	C1234568
ACCUMULATOR	nnnnnnnnnnnn	+08+01234568
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (2) Memory Decrement

```
OP      A
--     ---
57  A FIELD (SN)
```

	BEFORE	AFTER
A FIELD	+1234567	C1234566
ACCUMULATOR	nnnnnnnnnnnn	+08+01234566
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10 ARITHMETIC; FLOATING POINT, FIXED FIELD LENGTH

Fixed field length arithmetic instructions use a 20-digit accumulator which holds the instruction result within the processor as an operand for the next operation. Every instruction has an implied reference to the accumulator.

The floating point (real) format consists of a signed exponent field followed by a signed mantissa field. The representation of a floating point field is:

S/X, EXP, S/M, MANTISSA

Where:	S/X is the sign of the exponent (1 digit)
	EXP is the exponent (2 digits)
	S/M is the sign of the mantissa (1 digit)
	Mantissa is the mantissa data (8 or 16 digits)

Example: +01-87654321

The mantissa is a numeric field of two possible lengths; eight digit-Single Precision or sixteen digit-Double Precision. The mantissa is assumed to always have the decimal point to the left of the most significant digit.

The same accumulator is used for Fixed Point and Floating Point instructions.

If all sixteen digits of the accumulator mantissa are equal to zero, the exponent and mantissa sign will be set to -99+.

The Overflow Flag is set and the Comparison Flags are set to HIGH on overflow, to LOW on underflow and to EQUAL on a divide by zero.

1C ARITHMETIC FLOATING POINT, FIXED FIELD LENGTH (Continued)

A Trap Fault is a software enabled routine that allows the instruction in error to be examined.

The Trap Fault will be enabled if the two digit key stored at memory address 64, relative to Base #0, is equal to "FF".

If Trap Fault is enabled and a fault occurs, a Hardware Call procedure will be executed with the address of the instruction at fault stored on the stack.

If Trap Fault is not enabled and a fault occurs, the next program instruction will be executed.

A Trap Fault is caused when:

1. The resultant normalized mantissa is non-zero and the exponent is greater than +99 (overflow).

2. The resultant normalized mantissa is non-zero and the exponent is algebraically less than -99 (underflow).

3. The most significant digit of the mantissa of the divisor is equal to zero (divide by zero).

The result is not stored in memory. Division by zero does not change the contents of the accumulator. For all other instructions, the final contents of the accumulator will be unspecified.

10.1 REAL LOAD (RLD)/OP=78

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 78

A = Address of the source data field operand. Address may be indexed, indirect or extended. When the final address controller is equal to "0" or "2", the data field will be Single Precision. When the final address controller is equal to "1", the data field will be Double Precision.

Function

The real load instruction loads the accumulator with a floating point data field located in memory (A). The source data field is assumed to be in the form shown below:

```
SINGLE PRECISION
+--------+-----------+-----------+
|Sx Ex Sm|D1 D2 D3 D4|D5 D6 D7 D8|
+--------+-----------+-----------+

DOUBLE PRECISION
+--------+-----------+-----------+-------------+---------------+
|Sx Ex Sm|D1 D2 D3 D4|D5 D6 D7 D8|D9 D10 D11 D12|D13 D14 D15 D16|
+--------+-----------+-----------+-------------+---------------+
```

Sx is the sign of the exponent (1 digit). Ex is the exponent (2 digits). Sm is the sign of the mantissa (1 digit). D1-D8 is the single precision mantissa (8 digits) D1-D16 is the double precision mantissa (16 digits).

10.1 REAL LOAD (RLD)/OP=78 (Continued)

When the operation is single precision, only the eight most significant digits of the mantissa are loaded into the accumulator; the least significant eight digits are set to zero. When the operation is double precision all sixteen digits of the mantissa are loaded into the accumulator. The exponent and both signs will be loaded in the form that they appear in memory. Undigits may be loaded into the exponent and the mantissa fields of the accumulator. The result of loading undigits into the sign digit is machine dependent. See Appendix A - Compatibility Notes (A.54).

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

EXAMPLE (1) Load Accumulator with Floating Point Number

```
OP      A
--     ---
78  A FIELD (UN)
```

	BEFORE	AFTER
A FIELD	+07D12345678	unchanged
ACCUMULATOR	nnnnnnnnnnn nnnnnnnn	+07-12345678 00000000
COMPARISON	nnn	LOW

10.2 REAL STORE (RST)/OP=79

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 79

A = Address of the destination field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision.. A final address controller value of "1" indicates Double Precision.

Function

The real store instruction will store in a memory location (A) the contents of the accumulator, including the exponent, its sign and the sign of the mantissa. Undigits may be stored from the exponent and mantissa fields of the accumulator. The handling of the sign digit is machine dependent. See Appendix A - Compatibility Notes (A.54).

If the operation is single precision, the least significant eight digits of the accumulator are ignored.

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

10.2 REAL STORE (RST)/OP=79 (Continued)

EXAMPLE (1) Store Accumulator in Memory in Floating Point Notation

```
OP       A
--      ---
79  A FIELD (UN)
```

	BEFORE	AFTER
A FIELD	nnnnnnnnnnnn	C07D12345678
ACCUMULATOR	+07-12345678 nnnnnnnn	unchanged
COMPARISON	nnn	LOW

10.3 REAL ADD (RAA)/OP=70

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 70

A = Address of the Addend field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real add instruction adds the floating point number stored in a memory location (A) to the value stored in the accumulator and stores the sum in the accumulator.

The initial and final value of the accumulator and memory will have the same precision.

Different machines may maintain differing number of significant digits while performing the computation, thereby producing slightly different results. See Appendix A - Compatibility Notes (A.18).

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). The result of the operation will always be normalized.

If the operand data contains undigits other than in the sign digits, cause an Invalid Arithmetic Data. See Appendix A - Compatibility Notes (A.16).

10.3 REAL ADD (RAA)/OP=70 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

EXAMPLE (1) Add Floating number to Accumulator

```
OP       A
--      ---
70   A FIELD (UN)
```

	BEFORE	AFTER
A FIELD	+05+22222222	unchanged
ACCUMULATOR	+05+12345678	+05+34567900
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.4 REAL ADD AND STORE (RAS)/OP=71

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 71

A = Address of the Addend and Sum field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real add and store instruction will add the floating point number stored in a memory location (A) to the value stored in the accumulator and store the sum in the accumulator and in the same memory location (A). The store to memory will not take place on an error condition.

The initial and final value of the accumulator and memory will have the same precision.

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). The result of the operation will always be normalized.

If the operand data contains undigits other than in the sign digits, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.4 REAL ADD AND STORE (RAS)/OP=71 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

. EXAMPLE (1) Add Floating Number to Accumulator and Store

OP	A
71	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+22222222	C05C34567900
ACCUMULATOR	+05+12345678	+05+34567900
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.5 REAL SUBTRACT (RSU)/OP=72

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 72

A = Address of the subtrahend field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real subtract instruction will subtract the floating point number stored in a memory location (A) from the value stored in the accumulator and store the difference in the accumulator.

The initial and final value of the accumulator and memory will have the same precision.

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). The result of the operation will always be normalized.

If the operand data contains undigits other than in the sign digits, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.5 REAL SUBTRACT (RSU)/OP=72 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

EXAMPLE (1) Subtract Floating Number from the Accumulator

OP	A
72	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+11111111	unchanged
ACCUMULATOR	+05+12345678	+04+12345670
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.6 REAL SUBTRACT AND STORE (RSS)/OP=73

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 73

A = Address of the subtrahend and difference field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real subtract and store instruction will subtract the floating point number stored in a memory location (A) from the value stored in the accumulator and store the difference in the accumulator and in the same memory location (A). The store to memory will not take place on an error condition.

The initial and final value of the accumulator and memory will have the same precision.

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). The result of the operation will always be normalized.

If the operand data contains undigits other than in the sign digits, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.6 REAL SUBTRACT AND STORE (RSS)/OP=73 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

EXAMPLE (1) Subtract Floating Number from the Accumulator and Store

OP	A
73	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+11111111	C04C12345670
ACCUMULATOR	+05+12345678	+04+12345670
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.7 REAL MULTIPLY (RMU)/OP=74

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 74

A = Address of the multiplier field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real multiply instruction multiplies the value stored in the accumulator by the floating point number stored in a memory location (A) and stores the product in the accumulator.

The initial and final value of the accumulator will be 16 digits regardless of whether the input was single or double precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). If the input operands are normalized, the result will be normalized, and if they are not normalized, they may not produce a normalized result.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.7 REAL MULTIPLY (RMU)/OP=74 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

EXAMPLE (1) Multiply Accumulator by Floating Point Number

OP	A
74	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+30000000	unchanged
ACCUMULATOR	+05+12345678	+09+37037034
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.8 REAL MULTIPLY AND STORE (RMS)/OP=75

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 75

A = Address of the multiplier and product field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real multiply and store instruction multiplies the value stored in the accumulator by the number stored in a memory location (A) and stores the product in the accumulator and in the same memory location (A). The store to memory will not take place on an error condition.

The initial and final value of the accumulator will be 16 digits regardless of whether the input was single or double precision.

The operands need not be normalized, but incompatible results may be produced. See Appendix A - Compatibility Notes (A.17). If the input operands are normalized, the result will be normalized, and if they are not normalized, they may not produce a normalized result.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.8 REAL MULTIPLY AND STORE (RMS)/OP=75 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

Examples

EXAMPLE (1) Multiply Accumulator by Floating Number and Store

OP	A
75	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+30000000	C09C37037034
ACCUMULATOR	+05+12345678	+09+37037034
	nnnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.9 REAL DIVIDE (RDV)/OP=76

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 76

A = Address of the divisor field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "2" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real divide instruction divides the value stored in the accumulator by the floating point number stored in a memory location (A) and stores the quotient in the accumulator.

The initial and final value of the accumulator and memory will have the same precision.

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands must be normalized.

Operands that are not normalized will be treated as being equal to zero.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.9 REAL DIVIDE (RDV)/OP=76 (Continued)

Comparison Flags

Division by zero is an error condition that will terminate the instruction, set the Overflow Flag, and set the Comparison Flags to EQUAL. Otherwise, set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

EXAMPLE (1) Divide Accumulator by Floating Point Number

OP	A
76	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+20000000	unchanged
ACCUMULATOR	+05+12345678	+00+61728390
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.10 REAL DIVIDE AND STORE (RDS)/OP=77

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 77

A = Address of the divisor/quotient field operand. Address may be indexed, indirect or extended. A final address controller value of "0" or "1" indicates Single Precision. A final address controller value of "1" indicates Double Precision.

Function

The real divide and store instruction divides the value stored in the accumulator by the floating point number stored in a memory location (A) and stores the quotient in the accumulator and in the same memory location (A). The store to memory will not take place on an error condition.

The initial and final value of the accumulator and memory will have the same precision.

Set the least significant eight digits of the accumulator to zero when the operation is single precision.

The operands must be normalized.

Operands that are not normalized will be treated as being equal to zero.

If the operand data contains undigits other than in the sign digit, cause an Invalid Arithmetic Data fault. See Appendix A - Compatibility Notes (A.16).

10.10 REAL DIVIDE AND STORE (RDS)/OP=77 (Continued)

Comparison Flags

Division by zero is an error condition that will terminate the instruction, set the Overflow Flag, and set the Comparison Flags to EQUAL. Otherwise, set the Comparison Flags to HIGH if the result is positive, EQUAL if the result is zero and LOW if the result is negative.

EXAMPLE (1) Divide Accumulator by Floating Point Number and Store

OP	A
77	A FIELD (UN)

	BEFORE	AFTER
A FIELD	+05+20000000	C00C61728390
ACCUMULATOR	+05+12345678	+00+61728390
	nnnnnnnn	00000000
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

10.11 ACCUMULATOR MANIPULATE (ACM)/OP=84

Format

```
+----+----+
| OP | AF |
+----+----+
```

OP = 84

AF = Operation variants.

Function

The accumulator manipulate instruction modifies the contents of the accumulator as specified by the AF variants.

All variants of this instruction reference the entire accumulator without regard to data type or precision.

AF = 0x, Normalize Accumulator (x = unused)

If the most significant digit of the mantissa is zero, the entire mantissa will be shifted left and the exponent will be decremented by one. Continue shifting until the leading digit is non-zero, set the Comparison Flags according to the mantissa sign. If all sixteen digits are zero, set the exponent and mantissa signs to -99+ and set the Comparison Flags to EQUAL. If there are no leading zeros, set the Comparison Flags according to the mantissa sign. If the resulting exponent is smaller than -99, set the Overflow Flag and set the Comparison Flags to LOW (underflow). If an underflow is produced and Trap is enabled, a Trap Fault will occur.

AF = 1x, Convert Floating Point to Fixed Point (x = unused)

The accumulator mantissa is assumed to be normalized.

1C.11 ACCUMULATOR MANIPULATE (ACM)/OP=84 (Continued)

The data in the accumulator is converted from floating point to fixed point format. The mantissa is shifted right and the exponent is incremented until equal to +08. Set the Comparison Flags according to the mantissa sign. If the original exponent is equal to or greater than +08, an overflow is produced. The conversion does not take place if it would cause an overflow. If the eight most significant digits are now equal to zero, set the exponent and signs to -99+, set the eight least significant digits to zero, set the Comparison Flags to EQUAL. If an overflow is produced and Trap is enabled, a Trap Fault will occur.

AF = 2x, Set the Mantissa Sign to Plus (+) (x = unused)

Set the mantissa sign to plus. If the mantissa is 0, set the Comparison Flags to EQUAL. If the mantissa is non-zero, set the Comparison Flags to HIGH.

AF = 3x, Set the Mantissa Sign to Minus (-) (x = unused)

If the mantissa is non-zero, set the mantissa sign to minus and set the Comparison Flags to LOW. If the mantissa is zero, set the mantissa sign to plus and set the Comparison Flags to EQUAL.

AF = 4x, Complement the Mantissa Sign (x = unused)

If the mantissa is non-zero, complement the mantissa sign. Set the Comparison Flags to HIGH if the sign is set plus and LOW if the sign is set minus. If the mantissa is zero, set the mantissa sign to plus and set the Comparison Flags to EQUAL.

AF = 5x, Zero the Accumulator (Set to -99+0) (x = unused)

Set the 16 digit accumulator mantissa to zero. Set the signed exponent field to -99 and set the mantissa sign to plus. Set the Comparison Flags to EQUAL.

10.11 ACCUMULATOR MANIPULATE (ACM)/OP=84 (Continued)

AF = 6n, Increment the Exponent by n (n = 0-9)

Increment the exponent by n. Set the Comparison Flags to HIGH if the mantissa sign is plus, LOW if minus and EQUAL if the mantissa is zero. Attempts to increment the exponent beyond +99 will cause an overflow. A Trap Fault will occur if Trap is enabled.

AF = 7n, Decrement the Exponent by n (n = 0-9)

Decrement the exponent by n. Set the Comparison Flags to HIGH if the mantissa sign is plus, LOW if minus, and EQUAL if the mantissa is zero. Attempts to decrement the exponent beyond -99 will cause an underflow. A Trap Event will occur if Trap is enabled.

All other variants are reserved and will cause an Invalid Instruction fault (IEX = 25) and terminate the instruction with no change to the accumulator.

Examples

EXAMPLE (1) Normalize Accumulator

OP	AF
84	00

	BEFORE	AFTER
ACCUMULATOR	+05+00123456 78901234	+03+12345678 90123400
COMPARISON	nnn	HIGH

10.11 ACCUMULATOR MANIPULATE (ACM)/OP=84 (Continued)

EXAMPLE (2) Convert Floating Point Number to Fixed Point Number

```
OP    AF
--    --
84    10

                  BEFORE           AFTER

ACCUMULATOR   +06+12345678     +08+00123456
                  90123456         00000000

COMPARISON         nnn              HIGH
```

EXAMPLE (3) Set Mantissa Sign to Plus

```
OP    AF
--    --
84    20

                  BEFORE           AFTER

ACCUMULATOR   -05-12345678     -05+12345678
                  90123456         90123456

COMPARISON         nnn              HIGH
```

EXAMPLE (4) Set Mantissa Sign to Minus

```
OP    AF
--    --
84    30

                  BEFORE           AFTER

ACCUMULATOR   +05+12345678     +05-12345678
                  90123456         90123456

COMPARISON         nnn              LOW
```

10.11 ACCUMULATOR MANIPULATE (ACM)/OP=84 (Continued)

EXAMPLE (5) Complement Mantissa Sign

```
OP   AF
--   --
84   40

                     BEFORE            AFTER

ACCUMULATOR    +05-12345678      +05+12345678
                   90123456          90123456

COMPARISON           nnn               HIGH
```

EXAMPLE (6) Clear Accumulator

```
OP   AF
--   --
84   50

                       BEFORE            AFTER

ACCUMULATOR    nnnnnnnnnnnn       -99+00000000
                   nnnnnnnn           00000000

COMPARISON             nnn             EQUAL
```

TC.11 ACCUMULATOR MANIPULATE (ACM)/OP=84 (Continued)

EXAMPLE (7) Increment Exponent by 4

OP	AF
84	64

```
                 BEFORE          AFTER

ACCUMULATOR   +05-12345678    +09-12345678
                 90123456        90123456

COMPARISON         nnn             LOW
```

EXAMPLE (8) Decrement exponent by 2

OP	AF
84	72

```
                 BEFORE          AFTER

ACCUMULATOR   +09+12345678    +07+12345678
                 90123456        90123456

COMPARISON         nnn             HIGH
```

11 ADDRESS BRANCHING

11.1 BRANCH/OP=2x

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 20, 21, 22, 23, 24, 25, 26, 27, 28, 2A, 2B

A = Branch Address. Address may be indexed, indirect or extended. When not extended the final address controller bits specify the most significant digit of the address. This permits branching to any address up to and including 299,998, relative to Base #1, without indexing or extension. When the address is indexed, the final Base Indicant should resolve to a value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments.

Function

If the condition specified for the branch is true or if the branch is unconditional, the "A" address is selected as the next program instruction address. If the condition specified is not true or if the instruction is a "NO-OP", the next instruction is fetched with no significant action. The address field of a non-taken branch or a "NOP" must have the same attributes as any address, however, the address does not have to resolve into a valid address. For example, a six digit address may not contain an undigit in the extended digit position as this condition causes the processor to mistakingly treat this address as an eight digit address. An odd address only causes errors if the branch is taken. Undigits in address positions other than the address controller and extended digit may result in incompatible behavior (See Appendix A, Compatibility Notes, A.52). The Comparison and Overflow Flags define the branch conditions.

11.1 BRANCH (Continued)

Note: Use of "branch prediction op codes" may result in incompatible behavior. See Appendix A, Compatibility Notes (A.06).

OP = 20 (NOP) NO OPERATION

This instruction performs no significant action.

OP = 21 (LSS) BRANCH ON LESS THAN CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are set LOW (COML= 1, COMH=0).

OP = 22 (EQL) BRANCH ON EQUAL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are set EQUAL (COML=1, COMH=1).

OP = 23 (LEQ) BRANCH ON LESS THAN or EQUAL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are set LOW or EQUAL (COML=1).

OP = 24 (GTR) BRANCH ON GREATER THAN CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are set HIGH (COML=0, COMH=1).

OP = 25 (NEQ) BRANCH ON NOT EQUAL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are not set EQUAL or cleared (COML=0 or COMH=0).

OP = 26 (GEQ) BRANCH ON GREATER THAN or EQUAL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are set HIGH or EQUAL (COMH=1).

11.1 BRANCH (Continued)

OP = 27 (BUN) BRANCH UNCONDITIONAL

This instruction always causes a branch to the "A" address.

OP = 28 (OFL) BRANCH ON OVERFLOW CONDITION

This instruction causes a branch to the "A" address if the Overflow Flag is set and resets the Overflow Flag.

OP = 2A (NUL) BRANCH ON NULL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are reset (COML=0 and COMH=0).

OP = 2B (GTN) BRANCH ON GREATER OR NULL CONDITION

This instruction causes a branch to the "A" address if the Comparison Flags are reset or set HIGH (COML=0).

Comparison Flags

The Comparison Flags are not altered by these instructions.

12 HALTS

Halt instructions are dependent upon an execution digit located in absolute memory location 48 which determines the course of action as follows:

ACTION	VALUE
Reserved	A - F
Halt is ignored.	3
Halt is executed if not Privileged Mode.	2
Halt is executed if Privileged Mode.	1
Halt is executed.	0

If the halt is executed, the processor will enter a WAIT state that requires operator intervention to allow the processor to continue the execution of the instruction.

If the Halt is not executed, no significant action will be performed by the processor.

12.1 HALT BRANCH (HBR)/OP=29

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 29

A = Branch Address. Address may be indexed, indirect or extended. When not extended the final address controller bits specify the most significant digit of the address. This permits branching to any address up to and including 299,998, relative to Base #1, without indexing or extension. When the address is indexed, the final Base Indicant should resolve to a value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments.

Function

The Halt Branch instruction conditionally executes a halt according to the halt digit in absolute address 48. See Appendix A - Compatibility Notes (A.24).

If the halt is to take place, operator intervention is required to continue. Once continued, instruction execution resumes at the final "A" address.

If the halt is to be ignored, instruction execution continues at the final "A" address.

Comparison Flags

The Comparison Flags are not affected by this instruction.

12.2 HALT BREAKPOINT (HBK)/OP=48

Format

```
+----+----+----+
| OP | AF | BF |
+----+----+----+
```

OP = 48

AF = Final AF value is ignored, but useful in identifying the specific HBK. May specify Indirect Field Length. However, the specification of an AF indirect field length may produce incompatible results. See Appendix A - Compatibility Notes (A.25).

BF = Eight bit breakpoint control mask. The field will not be recognized as indirect.

Function

The Halt Breakpoint instruction performs a mask test against a halt character in memory location 46 relative to Base #0. If a bit is set in the halt character that corresponds to a bit set in the mask, the instruction executes a halt according to the halt digit in absolute memory address 48. See Appendix A - Compatibility Notes (A.24).

If there is no correspondence between the bits in the mask and the bits in the breakpoint bit pattern, the next instruction is selected in normal sequence with no other significant action.

Comparison Flags

The Comparison Flags are unchanged.

Overlap

There are no overlap restrictions for this instruction.

13 ENVIRONMENT CHANGE

13.1 BRANCH COMMUNICATE (BCT)/OP=30

Format

```
+----+--------+
| OP |  AFBF  |
+----+--------+
```

OP = 30

AFBF = Function Address consisting of the low order four digits of an address that is relative to the specified Task's MCP Data Area. The high order digits are equal to zero. Indirect Field lengths may be specified.

Function

The Branch Communicate instruction is used to allow a user program to enter a function in the MCP environment. It stores processor state and registers in Hyper Call Stack Frame format on the stack of the called environment and passes control to the specified function.

Its function is similar to the function of the Hyper Call instruction except for the selection of the Function entry and the inability to directly pass parameters.

The following operations are performed by this instruction:

1. The four digit Function Address (AFBF) is used as an offset relative to the task's MCP data area to select a Function entry.

13.1 BRANCH COMMUNICATE (BCT)/OP=30 (Continued)

Each Function entry contains the following information:

INFORMATION	DIGITS
Environment Number	00-05
Next Instruction Address	06-11
Protection Field (DD)	12-13
Reserved	14-15
Interrupt Mask	16-17
Mode Indicators	18-19

Note - The lowest memory address = 00

If the Protection Field is not equal to "DD", cause an Invalid Instruction fault (IEX = 37) and terminate the instruction with no further action.

2. Resolve the Environment Number, contained in the Function entry, to point to the selected Environment Table entry. However, retain the Active Environment Number so that it may be stored on the stack.

   Resolve entry #0 of the Memory Area Table for the new environment from memory and maintain addressability along with Base #0 of the current environment. See Section 5.7 entitled "Resolving a Memory Area Table Entry".

13.1 BRANCH COMMUNICATE (BCT)/OP=30 (Continued)

3. The top of stack pointer, located at address 40 (Relative to the new environment's Base #0), is used as the starting address, relative to the new environment's Base #0, to store the Hyper Call Stack Frame.

The sum of the top of stack pointer and the size of the Hyper Call Stack Frame (96) and the size of the Hardware Call Stack Frame area (500) is compared to Limit #0. If the sum is equal to or greater than Limit #0, cause a Stack Overflow fault and terminate the instruction with no further action. Otherwise, store the Hyper Call Stack Frame in the following sequence.

		INFORMATION	DIGITS
Old TOS	==>	Accumulator	00-27
		Measurement Register	28-35
		Interrupt Mask	36-37
		Mobile Index Registers	38-69
		Mode Indicators	70-71
		COM & OVF Flags	72-73
		Active Environment Number	74-79
New IX3	==>	Next Instruction Address	80-85
		Saved IX3 Value	86-93
		Stack Frame Indicator (FE)	94-95
		Stack Parameters (0 to 9999 bytes)	
New TOS	==>		

Note - The lowest memory address = 00

4. Store the new address of the next available stack location (Relative to the new environment's Base #0), into memory location 40 (Relative to the new environment's Base #0).

13.1 BRANCH COMMUNICATE (BCT)/OP=30 (Continued)

5. Set the two most significant digits of IX3 to "C0" and set the six least significant digits of IX3 to the initial address specified in memory location 40 (Relative to the new environment's Base #0) plus 80.

   IX3 will now point to the Next Instruction Address in the Hyper Call Stack Frame.

6. Set the machine "state" as follows:

INFORMATION	SET TO
Next Instruction Address	Function Table
Active Environment Number	Function Table
Interrupt Mask	Function Table
Mode Indicators	Function Table
Measurement Register (user field)	000000
Comparison & Overflow Flags	RESET

7. Set the MOPOK line to "zero" while the Measurement register is being changed and set it to a "one" at all other times.

8. If Soft Fault is now enabled, examine the memory location specified by the Reinstate List entry pointer plus 8. If it is not equal to zero, execute a Hardware Call procedure that will store the address of the next instruction to be executed.

9. Load the Memory Area Table pointed to by the Active Environment Number.

10. Using the new Base/Limit information, resolve the next instruction address, relative to Base #1, and execute an unconditional branch to that address.

The use of the Mobile Index Registers or the Accumulator to pass parameters is invalid. The contents are not guaranteed.

13.2 ENTER (NTR)/OP=31

Format

```
+----+--------+---------+------------------+
| OP |  AFBF  |    A    |    PARAMETERS    |
+----+--------+---------+------------------+
```

OP = 31

AFBF = Length, in bytes, of the Parameter field. The maximum number of bytes moved is 9,999. A value of 0000 will move no data. AF or BF may specify indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A.19.1).

A = Branch Address. Address may be indexed, indirect or extended. When not extended the final address controller bits specify the most significant digit of the address. This permits branching to any address up to and including 299,998, relative to Base #1, without indexing or extension. When the address is indexed, the final Base Indicant should resolve to a value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments. When indexed by IX3, the initial contents of IX3 are used.

PARAMETERS = Data field to be stored in the stack.

Function

The Enter instruction stores control information and parameters into a stack located in memory and executes an unconditional branch to the instruction at the "A" address.

A six digit address containing the stack location, relative to Base #0, is specified in memory location 40 relative to Base #0.

13.2 ENTER (NTR)/OP=31 (Continued)

The Enter Stack Frame is stored in the following sequence.

	INFORMATION	DIGITS
Old TOS ==>	Next Instruction Address	00-05
	Saved IX3 Value	06-13
	Reserved	14
	COM & OVF Flags	15
	Stack Parameters (0 to 9999 bytes)	
New TOS ==>		

Note - Lowest memory address = 00

"COM & OVF Flags" contain the following information:

INFORMATION	BIT
Reserved	3
Overflow Flag	2
Comparison Low Flag	1
Comparison High Flag	0

Parameters = AFBF bytes of data located after the "A" address field in the instruction.

Set the two most significant digits of IX3 to "C0". Set the contents of IX3 to the initial address specified in memory location 40 relative to Base #0.

Store the new value of the next available stack location, relative to Base #0, into memory location 40 relative to Base #0.

If the address to be stored into location 40 exceeds six digits, cause an Invalid Instruction fault (IEX = 04). See Appendix A - Compatibility Notes (A.19.2).

13.2 ENTER (NTR)/OP=31 (Continued)

Comparison Flags

Reset the Comparison and Overflow Flags.

Overlap

Undefined results will be produced if the stack area overlaps with the instruction or it's parameters.

Examples

```
 EXAMPLE (1)  Enter

 ADDRESS    OP AFBF   A     PARAMETERS
 -------    -- ----  ---    ----------
 003016     31 0003 020166   203010


            BEFORE        AFTER

 NI         003016        020166
 IX3       +0000010      C0001024
 0000040    001024        001046 Top of Stack
 0001024    nnnnnn        003034 Address Pointer
                                 After Parameters
                        +0000010 IX3 Value
                               0 Zero Digit
                               5 COM & OVF Flags
                          203010 Parameters


 COMPARISON  HIGH         CLEAR
 OVERFLOW     ON           OFF
```

13.3 EXIT (EXT)/OP=32

Format

```
+----+---------+
| OP |    A    |
+----+---------+
```

OP = 32

A = Return Address. Address may be indexed, indirect or extended. When not extended the final address controller bits specify the most significant digit of the address. This permits branching to any address up to and including 299,998, relative to Base #1, without indexing or extension. When the address is indexed, the final Base Indicant should resolve to a value of "1". The processor will always treat the resolved address as being relative to Base #1. The processor will not check for improper memory assignments. The normal return address is obtained by setting the "A" address to zero, indexing by IX3 and setting the address controller to indirect. When the address is indexed by IX3, the initial contents of IX3 are used.

Function

The Exit instruction reverses the actions of the Enter (OP = 31) instruction, thus accomplishing an exit from the stack.

The instruction restores the settings of the Overflow and Comparison Flags as specified by IX3 plus 15 if the digit at IX3 plus 14 is zero. If the digit at IX3 plus 14 contains a one, the flags will not be restored. All other values (2 to F) are reserved.

Copy the least significant six digits contained in IX3 to memory location 40 relative to Base #0.

Copy the eight digits at the location specified by IX3 plus 6 to IX3.

13.3 EXIT (EXT)/OP=32 (Continued)

Comparison Flags

If the digit at IX3 + 14 is zero, set the Comparison Flags according to the least significant 2 bits of the digit at IX3 + 15 See Section 13.2 (Enter OP = 31).

Overlap

There are no overlap restrictions for this instruction.

EXAMPLE (1) Exit the Stack

```
OP   A
--  ---
32 F00000
```

	BEFORE	AFTER	
NI	nnnnnn	003034	
IX3	+0001024	+0000010	
0000040	001046	001024	
0001024	003034	unchanged	STACK
	+0000010	unchanged	
	06	unchanged	
	203010	unchanged	
COMPARISON	nnn	LOW	
OVERFLOW	nnn	ON	

13.4 VIRTUAL ENTER (VEN)/OP=35

Format

```
+----+------+---------+---------+
| OP | AFBF |    A    |    B    |
+----+------+---------+---------+
```

OP = 35

AFBF = Length, in bytes, of the Parameter field. The maximum number of bytes moved is 9,999. A value of 0000 will move no data. Indirect field lengths may be specified. An AF literal of B1, B2 or B3 will be interpreted as a length of 1, 2 or 3 characters in the "A" location. All other literals will cause an Invalid Instruction fault (IEX = 22).

A = Address of the parameter data field operand. Address may be indexed, indirect or extended. The final address controller must equal UA or cause an Invalid Instruction fault (IEX = 03).

B = Address of the twenty digit Environment field. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

The Environment field contains the following information:

INFORMATION	DIGITS
Environment Number	00-05
Branch Address	06-11
Reserved	12-19

Note - Lowest memory address = 00

Function

The Virtual Enter instruction checks the stack limit then stores control information and parameters onto the users stack, located in memory, and executes an unconditional branch to a location specified by the environment field (B) using the specified Memory Area Table.

13.4 VIRTUAL ENTER (VEN)/OP=35 (Continued)

The following operations are performed by this instruction.

1. Check the Environment field (B) for either of these two cases:

   a. If the Reserved area of the Environment field is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

   b. If the Environment Number, contained in the Environment field (B), is equal to zero, store zeroes into the "Active Environment Number" field of the Virtual Enter Stack Frame. Otherwise, store the current Active Environment Number in the Virtual Enter Stack Frame.

2. The top of stack pointer, located at memory address 40 (relative to Base #0), is used as the starting address, relative to Base #0 to store the Virtual Enter Stack Frame.

   The sum of the top of stack pointer and the size of the Virtual Enter Stack Frame (30) and the amount of parameters (2 x AFBF) and the size of the Hardware Call Stack Frame area (500) is compared to Limit #0. If the sum is equal to or greater than Limit #0, cause a Stack Overflow fault and terminate the instruction with no further action. Otherwise, store the Virtual Enter Stack Frame in the following sequence.

		INFORMATION	DIGITS
Old TOS	==>	Measurement Register (User Part)	00-05
		COM & OVF Flags	06-07
		Active Environment Number	08-13
New IX3	==>	Next Instruction Address	14-19
		Saved IX3 Value	20-27
		Stack Frame Indicator ("FF")	28-29
		Stack Parameters (0 to 9999 bytes)	
New TOS	==>		

Note - Lowest memory address = 00

13.4 VIRTUAL ENTER (VEN)/OP=35 (Continued)

"COM & OVF Flags" contain the following information in the least significant digit. The other digit is reserved for future use and will equal zero.

INFORMATION	BIT
Reserved	3
Overflow Flag	2
Comparison Low Flag	1
Comparison High Flag	0

3. Move the parameters from a location in memory (A) to the stack.

4. Set the two most significant digits of IX3 to "C0", and set the six least significant digits of IX3 to the initial address specified at memory location 40 plus 14, relative to Base #0, to point to the Next Instruction Portion of the Virtual Enter Stack Frame.

5. Store the new value of the next available stack location, relative to Base #0, into memory location 40, relative to Base #0.

6. Reset the Comparison and Overflow Flags.

   The Mode Indicators, the Accumulator, the Measurement register, the Moble Index Registers and the Interrupt Mask register are not changed by this instruction.

7. If the Environment Number (B) is equal to zero, then this is a Local VEN, which does not require an environment change (i.e., the correct Base/Limit pairs are already resident in the processor). The active Environment Number remains unchanged. Skip the rest of this step.

   If the Environment Number (B) is not equal to zero, it replaces the Active Environment Number. Locate and load the Memory Area Table specified by this new Active Environment Number.

13.4 VIRTUAL ENTER (VEN)/OP=35 (Continued)

8. Execute an unconditional branch to the address, relative to Base #1, that is contained in the Branch Address portion of the Environment field (B).

The Active Environment Table and the Environment Table being entered must share the same Data Area (Base #0). The processor will not check for improper memory assignments.

13.5 HYPER CALL (HCL)/OP=62

Format

```
+----+------+---------+---------+
| OP | AFBF |    A    |    B    |
+----+------+---------+---------+
```

OP = 62

AFBF = Length, in bytes, of the Parameter field. The maximum number of bytes moved is 9,999. A value of 0000 will move no data. Indirect field lengths may be specified. An AF literal of B1, B2 or B3 will be interpreted as a length of 1, 2 or 3 characters in the "A" location. All other literals will cause an Invalid Instruction fault (IEX = 22).

A = Address of the parameter data field operand. Address may be indexed, indirect or extended. The final address controller must equal UA or cause an Invalid Instruction fault (IEX = 03).

B = Address of the four digit Function Number. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

Function

The Hyper Call instruction is used to enter a function in the MCP environment. The top of stack limit is checked then the processor registers, state and parameters are stored on the stack of the called environment and control is transfered to the specified function.

13.5 HYPER CALL (HCL)/OP=62 (Continued)

The following operations are peformed by this instruction:

1. Locate the six digit address, relative to the Task MCP Data Area, of the Hyper Call Function Table at memory address 87 relative to the Task MCP Data Area.

2. The four digit Function Number (B) is used as an array subscript into the Hyper Call Function Table. If the Function Number is not numeric, cause an Address Error fault (AEX = 34) and terminate the instruction with no further action. If the resultant address exceeds the six digit Hyper Call Function Limit, located at memory address 94 relative to the MCP Data Area, cause an Address Error fault (AEX = 02) and terminate the instruction with no further action.

   Each Function entry contains the following information:

INFORMATION	DIGITS
Environment Number	00-05
Next Instruction Address	06-11
Protection Field (DD)	12-13
Reserved	14-15
Interrupt Mask	16-17
Mode Indicators	18-19

   Note - The lowest memory address = 00

   If the Protection Field is not equal to "DD", cause an Invalid Instruction fault (IEX = 37) and terminate the instruction with no further action.

3. Resolve the Environment Number, contained in the Function entry, to point to the selected Environment Table entry. However, retain the Active Environment Number so that it may be stored on the stack.

   Resolve entry #0 of the Memory Area Table for the new environment from memory and maintain addressability along with Base #0 of the current environment. See Section 5.7 entitled "Resolving a Memory Area Table Entry".

13.5 HYPER CALL (HCL)/OP=62 (Continued)

4. The top of stack pointer, located at address 40 (Relative to the new environment's Base #0), is used as the starting address, relative to the new environment's Base #0, to store the Hyper Call Stack Frame.

   The sum of the top of stack pointer and the size of the Hyper Call Stack Frame (96) and the amount of parameters (2 x AFBF) and the size of the Hardware Call Stack Frame area (500) is compared to Limit #0. If the sum is equal to or greater than Limit #0, cause a Stack Overflow fault and terminate the instruction with no further action. Otherwise, store the Hyper Call Stack Frame in the following sequence.

		INFORMATION	DIGITS
Old TOS	==>	Accumulator	00-27
		Measurement Register	28-35
		Interrupt Mask	36-37
		Mobile Index Registers	38-69
		Mode Indicators	70-71
		COM & OVF Flags	72-73
		Active Environment Number	74-79
New IX3	==>	Next Instruction Address	80-85
		Saved IX3 Value	86-93
		Stack Frame Indicator (FE)	94-95
		Stack Parameters (0 to 9999 bytes)	
New TOS	==>		

   Note - The lowest memory address = 00

5. Move the Parameters, if any, from a location in memory (A) to the Hyper Call Stack Frame.

6. Store the new address of the next available stack location (Relative to the new environment's Base #0), into memory location 40 (Relative to the new environment's Base #0).

13.5 HYPER CALL (HCL)/OP=62 (Continued)

7. Set the two most significant digits of IX3 to "C0" and set the six least significant digits of IX3 to the initial address specified in memory location 40 (Relative to the new environment's Base #0) plus 80.

   IX3 will now point to the Next Instruction Address in the Hyper Call Stack Frame.

8. Set the machine "state" as follows:

INFORMATION	SET TO
Next Instruction Address	Function Table
Active Environment Number	Function Table
Interrupt Mask	Function Table
Mode Indicators	Function Table
Measurement Register (user field)	000000
Comparison & Overflow Flags	RESET

9. Set the MOPOK line to "zero" while the Measurement register is being changed and set it to a "one" at all other times.

10. If Soft Fault is now enabled, examine the memory location specified by the Reinstate List entry pointer plus 8. If it is not equal to zero, execute a Hardware Call procedure that will store the address of the next instruction to be executed.

11. Load the Memory Area Table pointed to by the Active Environment Number.

12. Using the new Base/Limit information, resolve the next instruction address, relative to Base #1, and execute an unconditional branch to that address.

The use of the Mobile Index Registers or the Accumulator to pass parameters is invalid. The contents are not guaranteed.

13.6 RETURN (RET)/OP=63

Format

```
+----+----+
| OP | AF |
+----+----+
```

OP = 63

AF = Unused and reserved.

Function

The Return instruction is a companion instruction to the Hyper Call (OP = 62), and Virtual Enter (OP = 35) instructions and the Hardware Call procedure. It reverses the action of the calling instruction or procedure by loading machine "state" from the current stack, restoring the user environment and executing an unconditional branch to the location specified in the Stack Frame as the Next Program Instruction.

The value of IX3 plus 14 represents the address, relative to Base #0, of the two digit Stack Frame Indicator. The information contained in this field indicates the type of calling procedure that stored the stack frame and the type of Return to be executed.

INFORMATION	INDICATOR
VIRTUAL ENTER/VIRTUAL EXIT	FF
HYPER CALL(BCT)/HYPER RETURN	FE
HARDWARE CALL/RETURN	FD

All other the Stack Frame Indicator values are invalid and will cause an Invalid Instruction fault (IEX = 37).

The parameter values stored in the stack are unchanged by this instruction and are not copied into any other area of memory.

13.6.1 VIRTUAL ENTER/VIRTUAL EXIT

1. IX3 minus 14 represents the address of the Virtual Enter Stack Frame relative to Base #0. The information in the Virtual Enter Stack Frame is used to replace the respective "state" in the machine.

   The Virtual Enter Stack Frame has been stored in the following sequence.

	INFORMATION	DIGITS
Old TOS ==>	Measurement Register (User Part)	00-05
	COM & OVF Flags	06-07
	Active Environment Number	08-13
New IX3 ==>	Next Instruction Address	14-19
	Saved IX3 Value	20-27
	Stack Frame Indicator ("FF")	28-29
	Stack Parameters (0 to 9999 bytes)	
New TOS ==>		

   Note - Lowest memory address = 00

2. Replace the address in memory address 40, relative to Base #0, with the value of IX3 minus 14, relative to Base #0. After the "state" is loaded, replace the contents of IX3 with the value of IX3 in the Virtual Enter Stack Frame.

3. The Mode Indicators, the Accumulator, and the Interrupt Mask Register are not changed by this variant.

4. If the Environment Number, contained in the stack frame, is equal to zero, the environment being returned to is the same environment that is specified by the Active Environment Number. Since the correct Base/Limit pairs are already resident within the processor, skip the rest of this step.

   If the Environment Number, contained in the stack frame, is not equal to zero, then this instruction is a non-local Virtual Exit, which requires that a new Memory Area Table must be loaded.

13.6.1 VIRTUAL ENTER/VIRTUAL EXIT (Continued)

If the first digit of the Environment Number is equal to a "D" and the processor is not in Privileged Mode, cause an Invalid Instruction fault (IEX = 31) and terminate the instruction with no further action.

Load the Memory Area Table pointed to by the Environment Number in the stack frame.

5. Resolve the next instruction address, relative to Base #1, and execute an unconditional branch to that address.

6. The current Memory Area Table and the Memory Area Table being entered must share the same Data Area (Base #0). The processor may or may not check for improper memory assignments.

13.6.2 HYPER CALL/HYPER RETURN HARDWARE CALL/RETURN

1. This variant may only be executed in Privileged Mode.

2. IX3 minus 80 represents the address of the Hyper Call or Hardware Call Stack Frame, relative to Base #0. The information in the Hyper Call/Hardware Call Stack Frame is used to replace the respective "state" in the machine.

   The Hyper Call/Hardware Call Stack Frame has been stored in the following sequence.

		INFORMATION	DIGITS
New TOS	==>	Accumulator	00-27
		Measurement Register	28-35
		Interrupt Mask	36-37
		Mobile Index Registers	38-69
		Mode Indicators	70-71
		COM & OVF Flags	72-73
		Active Environment Number	74-79
Old IX3	==>	Next Instruction Address	80-85
		Saved IX3 Value	86-93
		Stack Frame Indicator (FE/FD)	94-95
		Stack Parameters (0 to 9999 bytes)	
Old TOS	==>		

   Note - The lowest memory address = 00

3. Replace the address in memory address 40, relative to Base #0, with the value of IX3 minus 80, relative to Base #0. After the "state" is loaded, replace the contents of IX3 with the value of IX3 in the Hyper Call/Hardware Call Stack Frame.

4. Set the MOPOK line to "zero" while the Measurement register is being changed and set it to a "one" at all other times.

5. Load the Memory Area Table pointed to by the Environment Number in the Hyper Call/Hardware Call stack frame.

13.6.2 HYPER CALL/HYPER RETURN (Continued)
HARDWARE CALL/RETURN

6. Examine the Soft Fault and Trace Mode Fault Condition Indicators to determine if a Hardware Call Procedure should be executed. If Soft Fault is now enabled, examine the Soft Fault Pending Flag in the Reinstate List entry for this task. If it is not equal to zero, then a Soft Fault Condition exists.

   If this variant is a Hyper Return (Stack Frame Indicator = FE) and Trace Mode is enabled, then a Trace Fault Condition exists.

   If this variant is a Hardware Return (Stack Frame Indicator = FD), ignore the Trace Mode until the execution of the following instruction is complete.

7. If a Fault Condition has been found, execute a Hardware Call Procedure that will store the address of the next instruction to be executed and report all existing Fault Conditions.

   Otherwise using the new Base/Limit information, resolve the next instruction address, relative to Base #1, and execute an unconditional branch to that address.

13.7 ADJUST STACK POINTER (ASP)/OP=61

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 61

AF = A length of six (6) must be specified directly or as an indirect field length or a literal.

BF = Unused and reserved. May be specified as indirect.

A = Address of the increment field. Address may be indexed, indirect or extended. The final address controller must be UN or cause an Invalid Instruction fault (IEX = 03).

Function

The Adjust Stack Pointer instruction is used to increment the value of the Top of Stack Pointer (located at memory address 40, relative to Base #0) and to determine if there is sufficient space between the Top of Stack Pointer and Limit #0.

The sum of the increment value (A) and the value of the Top of Stack Pointer, located at memory address 40 relative to Base #0 and the value of Base #0 and the size of the Hardware Call Stack Frame area (500) is compared to Limit #0.

If the sum is equal to or greater than Limit #0, cause a Stack Overflow fault that stores the address of the failing instruction and terminate the instruction with no further action.

Otherwise, store the sum of the increment value (A) and the value of the Top of Stack Pointer into memory address 40, relative to Base #0.

13.8 INTERRUPT (INT)/OP=90

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 90

AF = Length of the "A" data field. May be indirect. A value of "00" indicates that there are no units to be moved. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Eight bit Kernel Request code. May be specified as indirect.

A = Address of the data field. Address may be indexed, indirect or extended. The final address controller must be UN or cause an Invalid Instruction fault (IEX = 03).

Function

The Interrupt instruction is used to initiate a transfer of the system environment to the MCP Kernel and to pass the information in BF and in the "A" operand, if specified.

1. Store "06" into absolute memory location 32 - 33.

2. Store the value of BF in absolute memory location 34 - 35.

3. If AF does not equal zero, store the "A" data field in absolute memory location 8000. If the value of AF exceeds 40, cause an Invalid Instruction fault (IEX = 25).

4. Cause an Interrupt procedure that will store the address of the next instruction to be executed.

This instruction may only be executed in Privileged Mode.

13.9 VIRTUAL BRANCH REINSTATE (BRV)/OP=93

Format

```
+----+----+
| OP | AF |
+----+----+
```

OP = 93

AF = Unused and reserved.

Function

The Virtual Branch Reinstate instruction is a companion instruction to the Interrupt procedure. The instruction restores the processor registers according to the contents of the Interrupt Frame and transfers control to the task specified by the address of the Reinstate List entry pointer contained in IX1.

The following operations are performed by this instruction to exit the MCP Kernel environment.

1. Use IX1 to locate the Reinstate List Entry for the new task to execute. This new task is now referred to as the Current Task.

2. Store the Task Number from the Reinstate List Entry for the current task into absolute memory addresses 82-85.

3. Reset Kernel Mode.

4. Load the Interrupt Frame from the Reinstate List Entry for this task.

5. Set the Task Timer to the value located at the Time Slice Remaining field in the Specified Reinstate List Entry.

6. Set the MOPOK line to "zero" while the Measurement register is being changed and set it to a "one" at all other times.

13.9 VIRTUAL BRANCH REINSTATE (BRV)/OP=93 (Continued)

7. If any processor detected faults exist, the Virtual Branch Reinstate has failed. Store:

   (a) the Fault indicators into the Failed Hardware Call R/D Area Field of the Reinstate List Entry for this task

   (b) "07" into the State Indicator Field in the Reinstate List entry for this task

   (c) "07" into absolute memory location 32 and cause an Instruction Interrupt to the MCP Kernel.

8. Load the Memory Area Table pointed to by the "Active Environment Number", which was loaded from the Interrupt Frame.

9. Examine the Soft Fault and Trace Fault Condition Indicators to determine if a Hardware Call Procedure should be executed. If Trace mode is now set, then a Trace Fault Condition exists.

   If Soft Fault is now enabled, examine the Soft Fault Pending Flag in the Reinstate List entry for this task. If it is not equal to zero, then a Soft Fault Condition exists.

10. If a Fault Condition has been found, execute a Hardware Call Procedure that will store the address of the next instruction to be executed and report all existing Fault Conditions.

    Otherwise, use the new Base/Limit information to resolve the next instruction address (Relative to Base #1) and execute an unconditional branch to that address.

This instruction may only be executed in Privileged Mode.

14 DATA MOVEMENT

14.1 MOVE DATA (MVD)/OP=08

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 08

AF = Forward/Backward variant. AF = 00 means move data FORWARD. Otherwise, move data BACKWARD. AF may be indirect. A literal flag will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A.20.3).

BF = Unused & reserved, but may be specified as an indirect field length.

A = Address of the source data field operand. Address may be indexed, indirect or extended. The final address controllers are ignored.

B = Starting address of the destination data field. Address may be indexed, indirect or extended. The address controllers are ignored.

C = End address of the destination data field. Address may be indexed, indirect or extended. The address controllers are ignored.

Note: A non-Mod 4 difference between the "B" and "C" addresses may produce incompatible results. See Appendix A - Compatibility Notes (A.20.1).

Note: The "B" and "C" addresses must both be relative to the same Memory Area. The processor will not check for improper memory assignments.

14.1 MOVE DATA (MVD)/OP=08 (Continued)

Function

The Move Data instruction moves data from the source data field to the destination data field starting with the "B" address and continuing until the "C" address.

Move Forward Variant

If AF = 00, a move forward takes place. The end "C" address must be greater than the starting "B" address, otherwise the instruction has no effect. Source field digits are moved to the destination field in an ascending manner until the "B" address is equal to or greater than the "C" address. No data is moved into the "C" address memory location.

Move Backward Variant

If AF = 01, a move backward takes place. The end "C" address must be less than the starting "B" address, otherwise the instruction has no effect. Source field digits are moved to the destination field in a descending order until the "B" address is equal to or less than the "C" address. No data is moved into the "B" address memory location. No data is moved from the "A" address memory location.

Comparison Flags

The Comparison and Overflow Flags are unchanged by this instruction.

Overlap

Partical overlap of "A" and "B" may produce incompatible results. See Appendix A - Compatibility Notes (A.20.2).

14.1 MOVE DATA (MVD)/OP=08 (Continued)

Examples

EXAMPLE (1) Forward Move

```
OP AF BF   A      B      C
-- -- --  ---    ---    ---
08 C0 00 001000 002000 002016

                BEFORE               AFTER

0001000   0123456789ABCDEF        unchanged
0002000   nnnnnnnnnnnnnnnn     0123456789ABCDEF
```

EXAMPLE (2) Backward Move

```
OP AF BF   A      B      C
-- -- --  ---    ---    ---
08 01 00 005010 006032 006020

                  BEFORE             AFTER

0004098       9876543210AB         unchanged
0006020       nnnnnnnnnnnn        9876543210AB
```

14.2 MOVE LINKS (MVL)/OP=09

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 09

AF = Length of all three operands. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "C" address controller). An indirect field length may be indicated. A literal flag will cause an Invalid Instruction (IEX = 21). See Appendix A - Compatibility Notes (A.21.1).

BF = Unused & reserved, but may be specified as an indirect field length.

A = Address of the "A" data field operand. Address may be indexed, indirect or extended. The final address controller must be equal to the "C" address controller.

B = Address of the "B" data field operand. Address may be indexed, indirect or extended. The final address controller must be equal to the "C" address controller.

C = Address of the "C" data field operand. Address may be indexed, indirect or extended. The address controller data type may be UN, SN, or UA.

If the three address controllers are not equal, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.21.3).

14.2 MOVE LINKS (MVL)/OP=09 (Continued)

Function

The Move Links instruction moves the number of units specified by AF in the following manner. The "C" field data is saved. The "A" field data is moved to the "C" data field. The "B" field data is moved to the "A" data field. The saved "C" field data is moved to the "B" data field.

Comparison Flags

The Comparison and Overflow Flags are unchanged by this instruction.

Overlap

Any total or partial overlap may produce incompatible results. See Appendix A - Compatibility Notes (A.21.2).

Examples

EXAMPLE (1) Move Numeric Fields

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
09 05 00, A FIELD (UN), B FIELD (UN), C FIELD (UN)
```

	BEFORE	AFTER
A FIELD	12345	67890
B FIELD	67890	87654
C FIELD	87654	12345

14.2 MOVE LINKS (MVL)/OP=09 (Continued)

EXAMPLE (2) Move Alpha Fields

```
OP AF BF        A              B              C
-- -- --       ---            ---            ---
09 03 00, A FIELD (UA), B FIELD (UA), C FIELD (UA)
```

	BEFORE	AFTER
A FIELD	XYZ	MNO
B FIELD	MNO	GHI
C FIELD	GHI	XYZ

EXAMPLE (3) Two Field Exchange

```
OP AF BF        A              B              C
-- -- --       ---            ---            ---
09 04 00, A FIELD (SN), B FIELD (SN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	+2386	+0004
B FIELD	+0004	+2386

14.3 MOVE ALPHA (MVA)/OP=10

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 10

AF = Length of "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate the A-syllable is a literal.

BF = Length of "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the source data field. Address may be indexed, indirect or extended. Data type may be UN, SN, or UA.

B = Address of the destination data field. Address may be indexed, indirect or extended. Data type may be UN, SN, or UA.

Function

The Move Alpha instruction moves digits or characters, depending on the address controllers, from one location in memory to another memory location, left justified.

AF>BF:

If the source data field is longer than the destination field (AF>BF), move the left most BF units from the "A" field to the "B" field, ignoring the remainder of the "A" field data, and set the Overflow Flag.

14.3 MOVE ALPHA (MVA)/OP=10 (Continued)

AF<BF:

If the source data field is shorter than the destination field (AF<BF), move the data and fill the destination data field with trailing zeros (SN or UN) or blanks (UA).

UA-UA:

When both address controllers specify 8-bit format (UA-UA), move each character.

UN-UN:

When both address controllers specify unsigned 4-bit format (UN-UN), move each digit.

SN-SN:

When both address controllers specify signed 4-bit format (SN-SN), move each digit and set the sign of the destination data field to the standard EBCDIC form (C = positive, D = negative) of the sign of the source data field. (Note: A negative zero remains a negative zero.)

UA-UN:

When the "A" and "B" address controllers specify UA and UN, respectively, only move the low order digit of each character in the source data field to the destination data field.

UA-SN:

When the "A" and "B" address controllers specify UA and SN, respectively, only move the low order digit of each character in the source data field to the destination data field. Set the sign of the destination data field to the standard EBCDIC sign for the interpreted value of the sign located in the most significant digit of the source data field.

14.3 MOVE ALPHA (MVA)/OP=10 (Continued)

UN-UA:

When the "A" and "B" address controllers specify UN and UA, respectively, move each digit and set the zone (high order) digit of the character to be stored in the destination data field to the standard EBCDIC numeric subset code (F).

UN-SN:

When the "A" and "B" address controllers specify UN and SN, respectively, move each digit and set the sign of the destination data field to the standard EBCDIC positive sign code (C).

SN-UN:

When the "A" and "B" address controllers specify SN and UN, respectively, move each digit and ignore the sign of the source data field except for setting the Comparison Flags.

SN-UA:

When the "A" and "B" address controllers specify SN and UA, respectively, move each digit and set the zone (high order digit) digit of the character to be written in the destination data field to the standard EBCDIC numeric subset code (F). The most significant digit of the destination field is set to the standard EBCDIC form of the sign of the source field.

Comparison Flags

Set the Comparison Flags to HIGH if the numeric digits moved from the source data field are non-zero and the sign of the source field is interpreted as positive.

Set the Comparison Flags to EQUAL if the numeric digits moved from the source data field are all zero.

Set the Comparison Flags to LOW if the numeric digits moved from the source data field are non-zero and the sign of the source field is interpreted as negative.

14.3 MOVE ALPHA (MVA)/OP=10 (Continued)

Overlap

When the "A" and "B" controllers indicate UA data, the field lengths are equal (AF=BF) and the value of the final "B" address is within the "A" data field (Address "A" to "A"+2 X AF), repeat the source data field between the "A" and "B" addresses throughout the destination data field.

Cases of overlapping "A" and "B", other than described above, may produce incompatible results. See Appendix A - Compatibility Notes (A.03).

Examples

EXAMPLE (1) Move Numeric Field to a Signed Numeric Field

```
OP AF BF       A               B
-- -- --      ---             ---
10 05 03, A FIELD (UN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	23511	unchanged
B FIELD	nnnnn	C235
COMPARISON	nnn	HIGH
OVERFLOW	nnn	ON

14.3 MOVE ALPHA (MVA)/OP=10 (Continued)

EXAMPLE (2) Move Signed Numeric Field to an Alpha Field

```
OP AF BF      A            B
-- -- --     ---          ---
10 03 03, A FIELD (SN), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	+823	unchanged
B FIELD	nnnn	C8F2F3
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (3) Move an Alpha Field to a Signed Numeric Field

```
OP AF BF      A            B
-- -- --     ---          ---
10 03 05, A FIELD (UA), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D4D5D6	unchanged
B FIELD	nnnnnn	D45600
COMPARISON	nnn	LOW
OVERFLOW	nnn	unchanged

14.3 MOVE ALPHA (MVA)/OP=10 (Continued)

EXAMPLE (4) Repeat First Character

```
OP AF BF       A               B
-- -- --      ---             ---
10 05 05, A FIELD (UA), A FIELD+2 (UA)
```

	BEFORE	AFTER
A FIELD	F0nnnnnnnnnn	F0F0F0F0F0F0
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

14.4 MOVE NUMERIC (MVN)/OP=11

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 11

AF = Length of "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate the A-syllable is a literal.

BF = Length of "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the source data field. Address may be indexed, indirect or extended. Data type may be UN, SN, or UA.

B = Address of the destination data field. Address may be indexed, indirect or extended. Data type may be UN, SN, or UA.

Function

The Move Numeric instruction moves digits or characters, depending on the address controllers, from one location in memory to another memory location, right justified.

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

AF<BF:

If the source data field is shorter than the destination field (AF<BF), the data is right justified in the destination field and padded with leading zero digits in the cases of UN and SN or zero characters (F0) in the case of UA.

AF>BF:

If the source data field is longer than the destination field (AF>BF), examine the high order numeric digits of the source data field for non-zero content. If these digits are non-zero, set the Overflow Flag and terminate the instruction with no further action. If these digits are zero, left truncate the source field and move the remainder of the field.

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

UA-UA:

When both address controllers specify 8-bit format (UA-UA), move the numeric portion of each character in the source data field to the destination data field with the zone digit set to the standard EBCDIC numeric subset code (F).

UN-UN:

When both address controllers specify unsigned 4-bit format (UN-UN), move each digit.

SN-SN:

When both address controllers specify signed 4-bit format (SN-SN), move each digit and set the sign of the destination data field to the standard EBCDIC form of the sign of source data field (C for positive, D for negative).

UA-UN:

When the "A" and "B" address controllers specify UA and UN, respectively, only move the low order digit of each character in the source data field to the destination data field.

UA-SN:

When the "A" and "B" address controllers specify UA and SN, respectively, only move the low order digit of each character in the source data field to the destination data field. Set the sign of the destination data field to the standard EBCDIC sign for the interpreted value of the sign located in the most significent digit of the source data field.

UN-UA:

When the "A" and "B" address controllers specify UN and UA, respectively, move each digit and set the zone (high order digit) digit of the character to be written in the destination data field to the standard EBCDIC numeric subset code (F).

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

UN-SN:

When the "A" and "B" address controllers specify UN and SN, respectively, move each digit and set the sign of the destination data field to the standard EBCDIC positive sign code (C).

SN-UN:

When the "A" and "B" address controllers specify SN and UN, respectively, move each digit and ignore the sign of the source data field except for setting the Comparison Flags.

SN-UA:

When the "A" and "B" address controllers specify SN and UA, respectively, move each digit and set the zone (high order digit) digit of the character to be written in the destination data field to the standard EBCDIC numeric subset code (F). Set the most significant digit of the destination field to the standard EBCDIC form of the sign of the source data field.

Comparison Flags

Set the Comparison Flags to HIGH if the numeric digits moved from the source data field are non-zero and the sign of the source field is interpreted as positive.

Set the Comparison Flags to EQUAL if the numeric digits moved from the source data field are all zero.

Set the Comparison Flags to LOW if the numeric digits moved from the source data field are non-zero and the sign of the source data field is interpreted as negative.

Note: Move Numeric UA-UA and UA-UN cause incompatible result in the final comparison flags. See Appendix A - Compatibility Notes (A.04.2).

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

Overlap

When the "A" and "B" controllers indicate UN data and AF=BF, and the final "B" address one greater than the final "A" address, repeat the first digit of the source data field throughout the destination data field.

Cases of overlapping "A" and "B", other than described above, may produce incompatible results. See Appendix A - Compatibility Notes (A.04).

Examples

EXAMPLE (1) Move Numeric Field to a Shorter Numeric

```
OP AF BF       A             B
-- -- --      ---           ---
11 05 03, A FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	00123	unchanged
B FIELD	nnnnn	123
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

EXAMPLE (2) Move Numeric Field to a Longer Numeric Field

```
OP AF BF       A              B
-- -- --      ---            ---
11 03 05, A FIELD (SN), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	+123	unchanged
B FIELD	nnnnnn	C00123
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (3) Move a Numeric Field with an Overflow Condition

```
OP AF BF       A              B
-- -- --      ---            ---
11 05 03, A FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	12300	unchanged
B FIELD	nnnnnn	unchanged
COMPARISON	nnn	unchanged
OVERFLOW	nnn	ON

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

EXAMPLE (4) Move an Alpha Field to an Alpha Field

```
OP AF BF        A               B
-- -- --       ---             ---
11 03 03, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C2D4E9	unchanged
B FIELD	nnnnnn	F2F4F9
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

EXAMPLE (5) Repeat First Digit

```
OP AF BF        A               B
-- -- --       ---             ---
11 05 05, A FIELD (UN), A FIELD+1 (UN)
```

	BEFORE	AFTER
A FIELD	6nnnnn	666666
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

14.4 MOVE NUMERIC (MVN)/OP=11 (Continued)

EXAMPLE (6) Move Alpha Field to Signed Field

```
OP AF BF        A              B
-- -- --       ---            ---
11 03 03, A FIELD (UA),  B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	D1C0F3	unchanged
B FIELD	nnnn	D103
COMPARISON	nnn	LOW
OVERFLOW	nnn	unchanged

EXAMPLE (7) Move Signed Field to Alpha Field

```
OP AF BF        A              B
-- -- --       ---            ---
11 03 03, A FIELD (SN),  B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	+123	unchanged
B FIELD	nnnnnn	C1F2F3
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

14.5 MOVE WORDS (MVW)/OP=12

Format

OP	AFBF	A	B

OP = 12

AFBF = Length of both operands. A value of "0000" is equal to a length of 10,000 4-digit "words" or 40,000 digits. AF or BF may be indirect. A literal flag will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A.07.1).

A = Address of the source data field. Address may be indexed, indirect or extended. Final address controllers are ignored.

B = Address of the destination data field. Address may be indexed, indirect or extended. Final address controllers are ignored.

Note: Use of non-mod 4 "A" or "B" addresses may produce incompatible results. See Appendix A - Compatibility Notes (A.07.2).

Function

The Move Words instruction moves the number of four digit "words" specified by the concatenation of AF and BF from the source data field in memory to the destination data field in memory.

The contents of the source data field are unchanged (unless "A" and "B" partially overlap).

14.5 MOVE WORDS (MVW)/OP=12 (Continued)

Comparison Flags

The Comparison Flags are unchanged by this instruction.

Overlap

When the final "B" address is less than the final "A" address and the fields partially overlap, the source data field will be shifted by that number of digits to the left. When the "B" data field partially overlaps the "A" data field and "B" is greater than "A", repeat the data from the "A" address to the "B" address throughout the destination data field.

The "B" data field may totally overlap the "A" data field.

14.5 MOVE WORDS (MVW)/OP=12 (Continued)

Examples

EXAMPLE (1) Move Eight Digits

```
OP AFBF        A              B
-- ----       ---            ---
12 0002     A FIELD,       B FIELD
```

	BEFORE	AFTER
A FIELD	01020304	01020304
B FIELD	nnnnnnnn	01020304
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

EXAMPLE (2) Repeat Data Field

```
OP AFBF        A              B
-- ----       ---            ---
12 0002   A FIELD (UN), A FIELD+4(UN)
```

	BEFORE	AFTER
A FIELD	0123nnnnnnnn	012301230123
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

14.6 MOVE WORDS AND CLEAR (MVC)/OP=13

Format

```
+----+--------+---------+---------+
| OP |  AFBF  |    A    |    B    |
+----+--------+---------+---------+
```

OP = 13

AFBF = Length of both operands. A value of "0000" is equal to a length of 10,000 4-digit "words" or 40,000 digits. AF or BF may be indirect. A literal flag will cause an Invalid Instruction fault (IEX = 21) See Appendix A - Compatibility Notes (A.07.1).

A = Address of the source data field. Address may be indexed, indirect or extended. Final address controllers are ignored.

B = Address of the destination data field. Address may be indexed, indirect or extended. Final address controllers are ignored.

Note: Use of non-Mod 4 "A" or "B" addresses may produce incompatible results. See Appendix A - Compatibility Notes (A.07.2).

Function

The Move Words and Clear instruction moves the number of four digit "words" specified by the concatenation of AF and BF from the source data field in memory to the destination data field. The contents of the source data field are set to zero.

Comparison Flags

The Comparison Flags are unchanged by this instruction.

14.6 MOVE WORDS AND CLEAR (MVC)/OP=13 (Continued)

Overlap

When the final "B" address is less than the final "A" address and the fields partially overlap, the source data field will be shifted by that number of digits to the left.

When the "B" data field partially overlaps the "A" data field and "B" is greater than "A", the data from the "A" address to the "B" address will be right justified in the destination data field and filled with leading zeros.

When the "B" data field totally overlaps the "A" data field, the "A" field is rewritten but not cleared.

14.6 MOVE WORDS AND CLEAR (MVC)/OP=13 (Continued)

Examples

EXAMPLE (1) Move Eight Digits and Clear the Source Field

```
OP AFBF       A              B
-- ----      ---            ---
13 0002  A FIELD (UN)  B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	F1F2F3F4	00000000
B FIELD	nnnnnnnn	F1F2F3F4
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

EXAMPLE (2) Justify Data Field

```
OP AFBF       A              B
-- ----      ---            ---
13 0002, A FIELD (UN), A FIELD+4(UN)
```

	BEFORE	AFTER
A FIELD	1605nnnnnnnn	000000001605
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

14.7 MOVE REPEAT (MVR)/OP=14

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 14

AF = Length of the "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate that the A-syllable is a literal.

BF = Number of repetitions. A value of "00" is equal to 100 repetitions. BF may be indirect.

A = Address of the source field. Address may be indexed, indirect or extended. The final address controller data type should specify UN or UA. An SN data type will be treated as UN.

B = Address of the destination field. Address may be indexed, indirect or extended. The final address controller data type should specify UN or UA. An SN data type will be treated as UN.

Function

The Move Repeat instruction moves AF number of digits or characters, depending on the address controllers, from the "A" field to the "B" field such that there are BF consecutive copies of the result in the "B" field.

When both address controllers specify 8-bit format (UA), move each character.

When both address controllers specify unsigned 4-bit format (UN), move each digit.

14.7 MOVE REPEAT (MVR)/OP=14 (Continued)

When the "A" and "B" address controllers specify UA and UN, respectively, only move the low order digit of each character in the source data field to the destination data field.

When the "A" and "B" address controllers specify UN and UA, respectively, move each digit and set the zone (high order digit) digit of the character in the destination data field to the standard EBCDIC numeric subset code (F).

Comparison Flags

The Comparison Flags are unchanged.

Overlap

"A" and "B" may totally overlap or may have matching type-address overlap (see 4.9.4).

Partial overlap of the "A" and "B" fields other than matching type-address overlap, may produce incompatible results. See Appendix A - Compatibility Notes (A.15).

14.7 MOVE REPEAT (MVR)/OP=14 (Continued)

Examples

EXAMPLE (1) Repeat a 3 digit Numeric Field 4 Times

```
OP AF BF       A             B
-- -- --      ---           ---
14 03 04, A FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	057	unchanged
B FIELD	nnnnnnnnnnnn	057057057057
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

EXAMPLE (2) Repeat a 3 Character Alpha Field Twice in a Numeric Field

```
OP AF BF       A             B
-- -- --      ---           ---
14 03 02, A FIELD (UA), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	D4D5D6	D4D5D6
B FIELD	nnnnnn	456456
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

14.8 TRANSLATE (TRN)/OP=15

Format

```
+----+--------+---------+---------+---------+
| OP |  AFBF  |    A    |    B    |    C    |
+----+--------+---------+---------+---------+
```

OP = 15

AFBF = Number of digits or characters to be translated. A value of "0000" is equal to a length of 10,000 units. AF or BF may be indirect. A literal flag will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A.14.1).

A = Address of the source field to be translated. The address may be indexed, indirect or extended. The final address controller data type may be UN, SN or UA.

B = Address of the translate table. Address may be indexed, indirect or extended. The final address controller data type is ignored. Some processors have some restrictions on this address. See Appendix A - Compatibility Notes (A.14.4).

C = Address of the destination field. The address may be indexed, indirect or extended. The final address controller must be UA or UN. Use of SN data type will cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.14.2).

14.8 TRANSLATE (TRN)/OP=15 (Continued)

Function

The Translate instruction substitutes a character from the translate table in the "B" field for each digit or character in the source ("A") field and moves the substituted character or the low order digit of that character to the destination ("C") field.

If the "A" field's data type is UN or SN, assume a EBCDIC numeric subset zone (F) before translation. If SN, the first digit (sign) is ignored.

If the final "C" address controller data type is UN, store only the digit portion of each translated character.

Each "A" field character itself is used to calculate an offset to the "B" address. The character found at "B" + offset is substituted for the "A" field character and moved to the "C" field.

14.8 TRANSLATE (TRN)/OP=15 (Continued)

The offset can be calculated by mapping the bits of the "A" field character to form a 3 digit number as shown in Figure 14.8-1.

Figure 14.8-1 Offset Calculation

```
                                         8 bits
     "A" field character:  +---+---+---+---+  +---+---+---+---+
                           | a | b | c | d |  | e | f | g | h |
                           +---+---+---+---+  +---+---+---+---+
                             |   |   |   |      |   |   |   |
                          .--'   |   |   |      |   |   |   '-----.
                          |   .--'   |   |      |   |   '-----.   |
                          |   |   .--'   |      |   |         |   |
                          |   |   |      '--.   |   |         |   |
     3 digit offset:      |   |   |         |   |   |         |   |
                    +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
                    | 0 | a | b | c | | 0 | d | e | f | | 0 | g | h | 0 |
                    +---+---+---+---+ +---+---+---+---+ +---+---+---+---+
```

Ex. "A" field character "$" (5B) produces an offset of 266.

A tabulation of offsets for all possible characters is shown in Figure 14.8-2.

14.8 TRANSLATE (TRN)/OP=15 (Continued)

Figure 14.8-2 Offset Tabulation

LSD> / MSD	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	000	002	004	006	010	012	014	016	020	022	024	026	030	032	034	036
1	040	042	044	046	050	052	054	056	060	062	064	066	070	072	074	076
2	100	102	104	106	110	112	114	116	120	122	124	126	130	132	134	136
3	140	142	144	146	150	152	154	156	160	162	164	166	170	172	174	176
4	200	202	204	206	210	212	214	216	220	222	224	226	230	232	234	236
5	240	242	244	246	250	252	254	256	260	262	264	266	270	272	274	276
6	300	302	304	306	310	312	314	316	320	322	324	326	330	332	334	336
7	340	342	344	346	350	352	354	356	360	362	364	366	370	372	374	376
8	400	402	404	406	410	412	414	416	420	422	424	426	430	432	434	436
9	440	442	444	446	450	452	454	456	460	462	464	466	470	472	474	476
A	500	502	504	506	510	512	514	516	520	522	524	526	530	532	534	536
B	540	542	544	546	550	552	554	556	560	562	564	566	570	572	574	576
C	600	602	604	606	610	612	614	616	620	622	624	626	630	632	634	636
D	640	642	644	646	650	652	654	656	660	662	664	666	670	672	674	676
E	700	702	704	706	710	712	714	716	720	722	724	726	730	732	734	736
F	740	742	744	746	750	752	754	756	760	762	764	766	770	772	774	776

Note: MSD, LSD is the "A" field character. The corresponding tabulation entry is the offset.

14.8 TRANSLATE (TRN)/OP=15 (Continued)

Overflow/Comparison Flags

The Overflow and Comparison Flags are not changed by this instruction.

Overlap

If the "A" and "C" data types are both UA or both UN, the "A" and "C" fields may totally overlap. All other forms of overlap may produce incompatible results. See Appendix A - Compatibility Notes (A.14.3).

Examples

EXAMPLE (1) Translate 1 Character

OP	AF	BF	A	B	C
15	00	01	A FIELD (UA)	B FIELD	C FIELD (UA)

	BEFORE	AFTER
A FIELD	C6	C6
B ADRS+614	7C	7C
C FIELD	nn	7C
COMPARISON	nnn	unchanged
OVERFLOW	nnn	unchanged

14.9 EDIT (EDT)/OP=49

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 49

AF = Not used as "A" field length. AF may be indirect or may indicate that the A-syllable is a literal.

BF = Number of eight bit edit-operators and in line literals in the "B-field". A value of "00" is equal to a length of 100 characters. BF may be indirect.

A = Address of the source field to be edited. Address may be indexed, indirect or extended. The address controller data type may be UN, SN , or UA.

B = Address of the edit-operator field. Address may be indexed, indirect or extended. The final address controller data type is ignored and treated as UA.

C = Address of the destination field. Address may be indexed, indirect or extended. The final address controller data type must be UN or UA. Use of SN data type will cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.13.1).

Function

The Edit instruction moves digits or characters (depending on the "A" address controller) from the "A" field to the "C" field under control of the edit-operators in the "B" field. Characters may be moved, inserted or deleted according to the edit-operators. Data movement and editing are stopped by the exhaustion of edit-operators in the "B" field.

Unconditionally reset the Overflow Flag.

14.9 EDIT (EDT)/OP=49 (Continued)

The source or "A" field is considered positive for unsigned numeric (UN) format. For unsigned alpha (UA), the most significant digit of the most significant character is interpreted as the sign. For signed numeric (SN), the most significant digit of the field is the sign (which is otherwise ignored).

If the "C" address controller is other than UA, only insert the low order digit of each character in the edit table into the destination data field. Therefore, whenever a blank (40) is specified, a zero will be inserted.

The edit instruction uses an edit table that is located in memory locations 48-63 relative to Base #0. This table may be initialized to any desired set of insertion characters.

The edit-operator field consists of a string of two-digit instructions. Each instruction is of the format MAv. The "M" digit is the operation code portion of the edit-operator. The "Av" digit is the variant position of the edit-operator. The various edit-operators are summarized in Figure 14.9-1 which is followed by a more detailed description.

14.9 EDIT (EDT)/OP=49 (Continued)

Figure 14.9-1 Edit-Operators

INSTRUCTION		VARIANT	
M	NAME	Av	ACTION
0	MOVE DIGIT	0 thru 9	T<== 1 (SIGNIFICANCE) MOVE Av + 1 DIGITS
1	MOVE CHARACTERS	0 thru 9	T<== 1 (SIGNIFICANCE) MOVE Av + 1 CHARACTERS

2	MOVE SUPPRESS	0 thru 9	IF T = 1, M <== 0
			IF T = 0, READ EACH A-DIGIT, THEN
			IF A-DIGIT=0/, M <== 0
			IF A-DIGIT=0, THEN
			IF Q = 0, INSERT BLANK
			IF Q = 1, INSERT TABLE ENTRY 2

3	INSERT UNCONDITIONALLY	0 - 7	INSERT TABLE ENTRY 0 - 7

		8	IF A = +, INSERT TABLE ENTRY 0
			IF A = -, INSERT TABLE ENTRY 1

		9	IF A = +, INSERT BLANK
			IF A = -, INSERT TABLE ENTRY 1

		A	IF A = +, INSERT TABLE ENTRY 0 IS A = -, INSERT BLANK
		B	INSERT NEXT B CHARACTER

4	INSERT ON PLUS	0 - B	IF A = +, M <== 3
			IF A = -, THEN
			IF Q = 0, INSERT BLANK
			IF Q = 1, INSERT TABLE ENTRY 2
			IF Av = B, SKIP NEXT B CHAR.

5	INSERT ON MINUS	0 - B	IF A = -, M <== 3
			IF A = +, THEN
			IF Q = 0, INSERT BLANK
			IF Q = 1, INSERT TABLE ENTRY 2
			IF Av = B, SKIP NEXT B CHAR.

14.9 EDIT (EDT)/OP=49 (Continued)

INSTRUCTION		VARIANT	
M	NAME	Av	ACTION

6	INSERT SUPPRESS	0 - B	IF T = 1, M <== 3
			IF T = 0, THEN
			  IF Q = 0, INSERT BLANK
			  IF Q = 1, INSERT TABLE ENTRY 2
			  IF Av = B, SKIP NEXT B CHAR.

7	INSERT FLOAT	0 - B	IF T = 1, MOVE ONE DIGIT
			 IF Av = B, SKIP NEXT B CHAR.
			IF T = 0, READ ONE A-DIGIT, THEN
			 IF A-DIGIT=0/, THEN, T <== 1,
			   IF Av = 0 - 7, THEN
			     INSERT TABLE ENTRY 0 - 7
			       MOVE ONE DIGIT
			   IF Av = 8 * A = +, THEN
			     INSERT TABLE ENTRY 0,
			       MOVE ONE DIGIT
			   IF Av = 8 * A = -, THEN
			     INSERT TABLE ENTRY 1,
			       MOVE ONE DIGIT
			   IF Av = 9 * A = +, THEN
			     INSERT BLANK,
			       MOVE ONE DIGIT
			   IF Av = 9 * A = -, THEN
			     INSERT TABLE ENTRY 1,
			       MOVE ONE DIGIT
			   IF Av = A * A = +, THEN
			     INSERT TABLE ENTRY 0,
			       MOVE ONE DIGIT
			   IF Av = A * A = -, THEN
			     INSERT BLANK,
			       MOVE ONE DIGIT
			   IF Av = B, THEN
			     INSERT NEXT B CHAR
			       MOVE ONE DIGIT
			 IF A-DIGIT =0, THEN
			  IF Q = 0, INSERT BLANK,
			  IF Q = 1, INSERT TABLE ENTRY 2
			   IF Av = B, SKIP NEXT B CHAR.

14.9 EDIT (EDT)/OP=49 (Continued)

INSTRUCTION		VARIANT	
M	NAME	Av	ACTION

8	END FLOAT	0 - B	IF T = 1, THEN
			IF Av = B/, NO OPERATION
			IF Av = B, SKIP NEXT B CHAR.
			IF T = 0, M <== 3

9	CONTROL	0 1 2 3	T <== 0 T <== 1 Q <== Q/ SKIP A DIGIT/CHARACTER

"T" denotes a Flag that is set to zero initially and is set to a one (significance) if a digit or character is moved from the source data field to the destination data field or if the CONTROL edit-op (MAv/=/91) is executed. If "T" is equal to one, zero suppression will be inhibited.

"Q" denotes a Flag that is set to zero initially. It is set to a one with the Control edit-op (MAv = 92) if a "check protect" or other character is to be repeated.

14.9 EDIT (EDT)/OP=49 (Continued)

M = 0, MOVE DIGIT (Av = 0-9):

Set "T" to one (significance).

When the "A" and "C" address controllers both specify 4-bit format (UN or SN), move "Av"+1 digits from the source data field to the destination data field.

When the "A" and "C" address controllers both specify 8-bit format (UA), move the numeric portion of "Av"+1 characters in the source data field to the destination data field and set the zone digit to the EBCDIC numeric subset code (F).

When the "A" and "C" address controllers specify UA and UN respectively, only move the numeric portion of "Av" +1 characters in the source data field to the destination data field.

When the "A" and "C" address controllers specify (UN or SN) and UA respectively, move "Av"+1 digits in the source data field to the destination data field and set the zone digit (high order digit) of each character to be stored to the EBCDIC numeric subset code (F).

M = 1, MOVE CHARACTER (Av = 0-9):

Set "T" to one (significance).

When the "A" and "C" address controllers both specify 4-bit format (UN or SN), move "Av"+1 digits from the source data field to the destination data field.

When the "A" and "C" address controllers both specify 8-bit format (UA), move "Av"+1 characters from the source data field, unchanged, to the destination data field.

14.9 EDIT (EDT)/OP=49 (Continued)

When the "A" and "C" address controllers specify UA and UN respectively, only move the numeric portion of "Av" +1 characters in the source data field to the destination data field.

When the "A" and "C" address controllers specify (UN or SN) and UA respectively, move "Av"+1 digits in the source data field to the destination data field and set the zone digit (high order digit) of each caracter to be stored to the EBCDIC numeric subset code (F).

M = 2, MOVE SUPPRESS (Av = 0-9):

If "T" equals one (significance), perform the operation move digit (M =0).

If "T" equals zero and the first source digit (or the low order digit of the first character) has a value of zero, and "Q" equals zero, insert a blank (40) into the destination data field; if "Q" equals one, insert the edit table value at Base #0+52 into the destination data field. "Av"+1 indicates the number of digits/characters to be examined.

If "T" equals zero and the first source digit (or the low order digit of the first character) has a value other than zero (significance), perform the operation Move Digit (M = 0).

14.9 EDIT (EDT)/OP=49 (Continued)

M = 3, INSERT UNCONDITIONALLY (Av = 0-9, A, or B):

If "Av" equals "0-7", insert a character from the edit table at Base #0+48+2Av into the destination data field.

If "Av" equals "8" and the sign of the source data field is positive (+), insert the edit table entry at Base #0+48 into the destination data field.

If "Av" equals "8" and the sign of the source data field is negative (-), insert the edit table entry at Base #0+50 will be inserted into the destination data field.

If "Av" equals "9" and the sign of the source data field is positive (+), insert a blank (40) into the destination data field.

If "Av" equals "9" and the sign of the source data field is negative (-), insert the edit table entry at Base #0+50 into the destination data field.

If "Av" equals "A" and the sign of the source data field is positive (+), insert the edit table entry at Base #0+48 into the destination data field.

If "Av" equals "A" and the sign of the source data field is negative (-), insert a blank (40) into the destination data field.

If "Av" equals "B", insert the next character in the edit-op into the destination data field.

14.9 EDIT (EDT)/OP=49 (Continued)

M = 4, INSERT ON PLUS (Av = 0-9, A, or B):

If the sign of the source data field is positive (+), perform the operation Insert Unconditionally (M = 3).

If the sign of the source data field is negative (-) and "Q" equals zero, insert a blank (40) into the destination data field.

If the sign of the source field is negative (-) and "Q" equals one, insert the edit table entry at Base #0+52 into the destination data field.

If the sign of the source field is negative (-) and if "Av" equals "B", skip the next character in the edit-op field. However, if there are no characters left to skip in the edit-op field, then cause an Invalid Instruction fault (IEX=07).

M = 5, INSERT ON MINUS (Av = 0-9, A, or B):

If the sign of the source data field is negative (-), perform the operation Insert Unconditionally (M = 3).

If the sign of the source data field is positive (+) and "Q" equals zero, insert a blank (40) into the destination data field.

If the sign of the source data field is positive (+) and "Q" equals one, insert the edit table entry at Base #0+52 into the destination data field.

If the sign of the source data field is positive (+) and if "Av" equals "B", skip the next character in the edit-op field. However, if there are no characters left to skip in the edit-op field, then cause an Invalid Instruction fault (IEX=07).

14.9 EDIT (EDT)/OP=49 (Continued)

M = 6, INSERT SUPPRESS (Av = 0-9 A, or B):

If "T" equals one (significance), perform the operation Insert Unconditionally (M = 3).

If "T" equals zero and "Q" equals zero, insert a blank (40) into the destination data field.

If "T" equals zero and "Q" equals one, insert a character from the edit table at Base #0+52 into the destination data field.

If "T" equals zero and "Av" equals "B", skip the next character in the edit-op field. However, if there are no characters left to skip in the edit-op field, then cause an Invalid Instruction fault (IEX=07).

M = 7, INSERT FLOAT (Av = 0-9, A, or B):

If "T" equals one (significance), perform the operation Move Digit (M = 0, Av = 0).

If "T" equals one (significance) and "Av" equals a "B", skip the next character in the in the edit-op field. However, if there are no characters left to skip in the edit-op field, then cause an Invalid Instruction fault (IEX=07).

If "T" equals zero and the source digit (AC=SN or UN) or the low order digit of the source character (AC=UA) has a value of zero and "Q" equals zero, insert a blank (40) into the destination data field. If "Q" equals a one, insert the edit table entry at Base #0+52 into the destination data field.

If "T" equals zero and the source digit (AC=SN or UN) or the low order digit of the source character (AC=UA) has a value of other than zero, perform the operation Insert Unconditionally (M = 3), set T to one and perform the operation Move Digit (M = 0, Av = 0). If Av = B, skip next character in edit-operator field. However, if there are no characters left to skip in the edit-op field, then cause an Invalid Instruction fault (IEX=07).

14.9 EDIT (EDT)/OP=49 (Continued)

M = 8, END FLOAT (Av = 0-9, A, or B):

If "T" equals one (significance) and "Av" is not equal to a "B", no operation is performed.

If "T" equals one (significance) and "Av" equals a "B", skip the next character in the in the edit-op field.

If "T" equals zero, perform the operation Insert Unconditionally (M = 3).

M = 9, CONTROL (Av = 0-3):

This edit-operator performs a control function based on the variant (Av).

Variant	Action
0	Set "T" to Zero
1	Set "T" to One
2	Complement "Q"
3	Skip the Source Data Field Digit or Character

Note: Use of undigits A-F for "M" or use of the values for "Av" not specified above will cause an Invalid Instrucrtion fault (IEX = 07). See Appendix A - Compatibility Notes (A.13.2).

14.9 EDIT (EDT)/OP=49 (Continued)

Overflow/Comparison Flags

Set the Comparison Flags to HIGH if the numeric digits moved from the source data field are non-zero and the sign of the source field is interpreted as positive.

Set the Comparison Flags to LOW if the numeric digits moved from the source data field are non-zero and the sign of the source field is interpreted as negative.

Set the Comparison Flags to EQUAL if all the numericsam digits moved from the source data field are equal to zero or if no character or digit is moved from the source data field.

Reset the Overflow flag.

Overlap

Overlap of the "A", "B", or "C" fields in any manner may produce incompatible results. See Appendix A - Compatibility Notes (A.13.3).

14.9 EDIT (EDT)/OP=49 (Continued)

Examples

```
EXAMPLE (1)  Edit

  OP AF BF        A            B            C
  -- -- --       ---          ---          ---
  49 00 01, A FIELD (UA), B FIELD (UA), C FIELD (UA)


A FIELD                 C1C2C3
B FIELD                     02
C FIELD (AFTER)         F1F2F3

COMPARISON (AFTER)        HIGH


EXAMPLE (2)  Edit

  OP   AF   BF          A                B                 C
  --   --   --         ---              ---               ---
  49   00   22,   A FIELD (SN),    B FIELD (UA),    C FIELD (UA)

A FIELD                                                   C0 01 30 59
B FIELD                           4B D7 4B C1 4B E8 37 92 75 64 75
                                  75 75 85 93 33 01 92 5B C3 5B D9
TABLE(48-62)                               4E 60 5C 4B 6B 5B F0 40
                                            +  -  *  .  ,  $  0  b
C FIELD (AFTER)       D7 C1 E8 40 5C 5C 5C 5B F1 F3 4B F5 F9 40 40
                       P  A  Y  b  *  *  *  $  1  3  .  5  9  b  b

Note: b = Blank

COMPARISON (AFTER)                                               HIGH
```

15 LOGICAL

15.1 SCAN TO DELIMITER - EQUAL (SDE)/OP=16

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 16

AF = Length of the "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate that the A-syllable is a literal.

BF = Length of the "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the delimiter list field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

B = Address of the data field to be scanned. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

15.1 SCAN TO DELIMITER - EQUAL (SDE)/OP=16 (Continued)

Function

The Scan to Delimiter-Equal instruction scans the "B" field for a character equal to one of the delimiter characters from the "A" field.

The first "B" field character is compared to each delimiter ("A") field character until a match is found. If no match is found, the next "B" field character is compared to each delimiter. Continue this process until a matching delimiter is found or until the "B" field is exhausted.

Note: If a numeric data type (UN/SN) is specified in either field, add the EBCDIC zone digit "F" to each digit to form the character for use in the comparison.

This instruction stores the number of characters in the "B" data field PRECEDING the equal character into memory locations 38-39, relative to Base #0 (accessible with indirect field length). However, if no equal character is found, store the field length of the data field minus one (BF - 1).

Comparison Flags

Set the Comparison Flags to HIGH, if no characters (in the "B" field) were found to be equal to any delimiter (in the "A" field), LOW if the first character was equal to any delimiter and EQUAL if any character but the first was equal to any delimiter.

Overlap

There are no overlap restrictions for this instruction.

15.1 SCAN TO DELIMITER - EQUAL (SDE)/OP=16 (Continued)

Examples

EXAMPLE (1) Scan Delimiter-Equal, First digit Equal

```
OP AF BF        A              B
-- -- --       ---            ---
16 01 04, A FIELD (UN), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	1	unchanged
B FIELD	F1C8C4D9	unchanged
0000038	nn	00
COMPARISON	nnn	LOW

EXAMPLE (2) Scan Delimiter-Equal, Other Than First digit equal

```
OP AF BF        A              B
-- -- --       ---            ---
16 01 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	E7	unchanged
B FIELD	C1C2E7F5	unchanged
0000038	nn	02
COMPARISON	nnn	EQUAL

15.2 SCAN TO DELIMITER - UNEQUAL (SDU)/OP=17

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 17

AF = Length of the "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate that the A-syllable is a literal.

BF = Length of the "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the delimiter list field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

B = Address of the data field to be scanned. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

Function

The Scan to Delimiter-Unequal instruction scans the "B" field for a character not equal to any of the delimiter characters from the "A" field.

The first "B" field character is compared to each delimiter ("A") field character until a match is found. If a match is found, the next "B" field character is compared to each delimiter. Continue this process until no matching delimiter is found for a given "B" field character or until the "B" field is exhausted.

15.2 SCAN TO DELIMITER - UNEQUAL (SDU)/OP=17 (Continued)

Note: If a numeric data type (UN/SN) is specified in either field, add the EBCDIC zone digit "F" to each digit to form the character for use in the comparison.

This instruction stores the number of characters in the data field PRECEDING the equal character in memory locations 38-39, relative to Base #0 (accessible with indirect field length). However, if no unequal character is found, store the field length of the data field minus one (BF - 1).

Comparison Flags

Set the Comparison Flags to HIGH if all the characters in the data field ("B") are equal to the characters in the delimiter list ("A"), LOW if the first character was not equal to any of the delimiters and EQUAL if some other character is unequal.

Overlap

There are no overlap restrictions for this instruction.

15.2 SCAN TO DELIMITER - UNEQUAL (SDU)/OP=17 (Continued)

Examples

EXAMPLE (1) Scan Delimiter-Unequal, First digit Unequal

```
OP AF BF        A               B
-- -- --       ---             ---
17 03 04, A FIELD (UN), B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	123	unchanged
B FIELD	6123	unchanged
0000038	nn	00
COMPARISON	nnn	LOW

EXAMPLE (2) Scan Delimiter-Unequal, Other Than First digit Unequal

```
OP AF BF        A               B
-- -- --       ---             ---
17 03 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1C2C3	unchanged
B FIELD	C1C2C3C4	unchanged
0000038	nn	03
COMPARISON	nnn	EQUAL

15.3 SCAN TO DELIMITER - ZONE EQUAL (SZE)/OP=18

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 18

AF = Length of the "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate that the A-syllable is a literal.

BF = Length of the "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the delimiter list field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

B = Address of the data field to be scanned. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

Function

The Scan to Delimiter-Zone Equal instruction scans the characters of the "B" field for a character whose zone digit is equal to the zone digit of any of the delimiter characters from the "A" field.

The zone digit of the first "B" field character is compared to the zone digit of each delimiter ("A") field character until a match is found. If no match is found, the next "B" field character's zone is compared to each delimiter's zone. Continue this process until a matching delimiter's zone is found or until the "B" field is exhausted.

15.3 SCAN TO DELIMITER - ZONE EQUAL (SZE)/OP=18 (Continued)

Note: If a numeric data type (UN/SN) is specified in either field, the EBCDIC zone digit "F" is used in the comparison.

This instruction stores the number of characters in the data field PRECEDING the zone-equal character in memory locations 38-39, relative to Base #0 (accessible with indirect field length). However, if no zone-equal character is found, store the field length of the data field minus one (BF - 1).

Comparison Flags

Set the Comparison Flags to HIGH if none of the zones of any of the data field characters are equal to the zone portion of any delimiter list character, LOW if the zone of the first data field character is equal to the zone of any delimiter field character, and EQUAL if the zone of any character but the first were equal.

Overlap

There are no overlap restrictions for this instruction.

15.3 SCAN TO DELIMITER - ZONE EQUAL (SZE)/OP=18 (Continued)

Examples

EXAMPLE (1) Scan Delimiter-Zone Equal, First Zone Equal

```
OP AF BF       A              B
-- -- --      ---            ---
18 02 03, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1D1	unchanged
B FIELD	D2E6C1	unchanged
0000038	nn	00
COMPARISON	nnn	LOW

EXAMPLE (2) Scan Delimiter-Zone Equal, Other Than First Zone Equal

```
OP AF BF       A              B
-- -- --      ---            ---
18 02 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1D1	unchanged
B FIELD	E6D2C1D4	unchanged
0000038	nn	01
COMPARISON	nnn	EQUAL

15.3 SCAN TO DELIMITER - ZONE EQUAL (SZE)/OP=18 (Continued)

EXAMPLE (3) Scan Delimiter-Zone equal, No Zones Equal

```
OP AF BF        A             B
-- -- --       ---           ---
18 04 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	F160C1D1	unchanged
B FIELD	E6E7E8E9	unchanged
0000038	nn	03
COMPARISON	nnn	HIGH

15.4 SCAN TO DELIMITER - ZONE UNEQUAL (SZU)/OP=19

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 19

AF = Length of the "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate that the A-syllable is a literal.

BF = Length of the "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the delimiter list field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

B = Address of the data field to be scanned. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA. An SN controller is treated as UN (eg. 7SN = 7UN).

Function

The Scan to Delimiter-Zone Unequal instruction scans the characters of the "B" field for a character whose zone digit is NOT equal to the zone digit of any of the delimiter characters from the "A" field.

The zone digit of the first "B" field character is compared to the zone digit of each delimeter ("A") field character until a match is found. If a match is found, the next "B" field character's zone is compared to each delimiter's zone. Continue this process until no matching delimiter's zone is found for a given "B" field character's zone or until the "B" field is exhausted.

15.4 SCAN TO DELIMITER - ZONE UNEQUAL (SZU)/OP=19 (Continued)

Note: If a numeric data type (UN/SN) is specified in either field, the EBCDIC zone digit "F" is used in the comparison.

This instruction stores the number of characters in the data field PRECEDING the zone-unequal character in memory locations 38-39, relative to Base #0 (accessible with indirect field length). However, if no zone-unequal character is found, store the field length of the data field minus one (BF - 1).

Comparison Flags

Set the Comparison Flags to HIGH if every data field zone matches a delimiter character zone, LOW if the zone of the first data field zone is not equal to the zone of any delimiter field character, and EQUAL if some zone, other than the first, in the data field is not equal to the zone of any delimiter list character.

Overlap

There are no overlap restrictions for this instruction.

15.4 SCAN TO DELIMITER - ZONE UNEQUAL (SZU)/OP=19 (Continued)

Examples

EXAMPLE (1) Scan Delimiter-Zone Unequal, First Zone Unequal

```
OP AF BF       A            B
-- -- --      ---          ---
19 01 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1	unchanged
B FIELD	D1C1C2E7	unchanged
0000038	nn	00
COMPARISON	nnn	LOW

EXAMPLE (2) Scan Delimiter-Zone Unequal, Other Than First Zone Unequal

```
OP AF BF       A            B
-- -- --      ---          ---
19 02 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1D1	unchanged
B FIELD	C1C2E7C3	unchanged
0000038	nn	02
COMPARISON	nnn	EQUAL

15.4 SCAN TO DELIMITER - ZONE UNEQUAL (SZU)/OP=19 (Continued)

EXAMPLE (3) Scan Delimiter-Zone Unequal, No Zones Unequal

```
OP AF BF       A              B
-- -- --      ---            ---
19 02 04, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1D1	unchanged
B FIELD	C3C4D4D6	unchanged
0000038	nn	03
COMPARISON	nnn	HIGH

15.5 SEARCH (SEA)/OP=39

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 39

AF = Number of units (digits or characters, depending on the "A" address controller) to be compared between the two data fields. A value of "00" is equal to a length of 100 units. AF may be indirect or may indicate a literal.

BF = Number of units (digits or characters, depending on the "B" address controller) that the Table Entry (B) will be incremented between comparisons. A value of "00" is equal to a length of 100 units. BF may be indirect.

A = Address of the key field. Address may be indexed, indirect or extended. The final address controllers specify the format for both the "A" and "B" fields and may be UN, SN or UA.

B = Address of the first table entry. Address may be indexed, indirect or extended. The data type is the same as that specified for the key field (A). The final address controller bits determine the incrementation between comparisons. The Base Indicant of the Table Entry (B) and Table Limit (C) addresses must be the same. The processor will not check for improper memory assignments.

B Address Controller	Increment in Digits
00 (UN)	BF
01 (SN)	BF+1
10 (UA)	2 x BF

15.5 SEARCH (SEA)/OP=39 (Continued)

C = Address of the table limit. Address may be indexed, indirect or extended. The final address controller bits specify the type of search to be performed. The Base Indicant of the Table Entry (B) and Table Limit (C) addresses must be the same. The processor will not check for improper memory assignments.

C Address Controller	Search Type
00 (UN)	Search for Equal
01 (SN)	Search for Low
10 (UA)	Search for Lowest

Function

The Search instruction compares the key field (A) with the first Table Entry (B) in the manner prescribed by the "C" address controller variants, then increments the Table Entry address by the amount specified by BF and the "B" address controller. This new location is compared with the key field (A). Continue this operation of compare and increment until the searched for condition is found or when the Table Entry is equal to or greater than the Table Limit (C), except in the case of "Search Lowest".

The Overflow Flag is reset by this instruction.

15.5 SEARCH (SEA)/OP=39 (Continued)

Search for Equal:

Terminate the search when a Table Entry field equal to the key field (A) is found, or when the Table Limit (C) is reached or exceeded.

If a Table Entry field equal to the key (A) is found, store the address of the Table Entry field, relative to the same base as the "B" operand in IX1 with the same Base Indicant as the resolved "B" operand. Otherwise, IX1 remains unchanged (Search for Equal Condition NOT met).

For SN data, a positive zero does not compare equal to a negative zero.

Search for Low:

Terminate the search when the first Table Entry field less than the key field (A) is found, or when the Table Limit is reached or exceeded.

If a Table Entry field less than the key (A) is found, store the address of that Table Entry field, relative to the same base as the "B" operand, in IX1 with the same Base Indicant as the resolved "B" operand. Otherwise, IX1 remains unchanged (Search for Low Condition NOT met).

For SN data, a negative zero compares less than a positive zero.

15.5 SEARCH (SEA)/OP=39 (Continued)

Search for Lowest:

Terminate the search only when the Table Entry field address reaches or exceeds the Table Limit (C). If any Table Entry fields are found that are less than the key (A), store the first Table Entry field, relative to the same base as the "B" operand, WHICH IS LESS THAN OR EQUAL TO ALL THOSE LESS THAN THE KEY in IX1 with the same Base Indicant as the resolved "B" operand. If NO Table Entry fields are found that are less than the key (A), store the base relative value of the key (A) in IX1 with the two most significant digits of IX1 set to "C0". (Search for Lowest Condition NOT met).

For SN data, a negative zero compares less than a positive zero.

In each type of search, if the Table Entry field being compared to the key (A) overlaps the Table Limit, incompatible results may be produced. See Appendix A - Compatibility Note (A.12)

Comparison Flags

When the searched for condition is met, set the Comparison Flags to EQUAL. If the searched for condition is NOT met, set the Comparison Flags to HIGH.

Overlap

There are no overlap restrictions for this instruction.

15.5 SEARCH (SEA)/OP=39 (Continued)

Examples

EXAMPLE (1) Search Equal

```
OP AF BF        A              B          C
-- -- --       ---            ---        ---
39 01 02, A FIELD (UA), 1000 (UA), 1020 (UN)
```

	BEFORE	AFTER
A FIELD	C1	unchanged
B FIELD	C1F1C2F2C3F3C4F2C5F1	unchanged
IX1	nnnnnnnn	+0001000
COMPARISON	nnn	EQUAL
OVERFLOW	nnn	OFF

EXAMPLE (2) Search Low, Condition Not Found

```
OP AF BF        A              B          C
-- -- --       ---            ---        ---
39 01 01, A FIELD (UN), 1000 (UN), 1010 (SN)
```

	BEFORE	AFTER
A FIELD	2	unchanged
B FIELD	3459876345	unchanged
IX1	nnnnnnnn	unchanged
COMPARISON	nnn	HIGH
OVERFLOW	nnn	OFF

15.5 SEARCH (SEA)/OP=39 (Continued)

EXAMPLE (3) Search Lowest

```
OP AF BF        A            B          C
-- -- --       ---          ---        ---
39 01 01, A FIELD (UA), 1000 (UA), 1020 (UA)
```

```
                          BEFORE             AFTER

A FIELD                     C5              unchanged
B FIELD           C5C2C3C4C9C3C1E2C3C9      unchanged
IX1                      nnnnnnnn           +0001012
                                      (Points to Letter A)

COMPARISON                 nnn               EQUAL
OVERFLOW                   nnn                OFF
```

15.6 SEARCH LINK LIST (SLL)/OP=37

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 37

AF = Length of the "A" data field. May be indirect or literal flag. A value of "00" is equal to a length of 100 units.

BF = Amount of offset in units from the "B" address to the field to be searched. BF is typically six digits or more to allow for the link address at "B". A value of "00" is equal to an offset of zero units. BF may be indirect.

A = Address of the key to which the "B" data field will be compared. Address may be indexed, indirect or extended. The final address controller specifies the format for both the "A" and "B" fields and must be UN or UA. If the address controller specifies SN, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.26.2).

B = Address of the first list entry. The initial address may be indexed or extended. Indirect addressing is not allowed. The data format is that of the final "A" address controller. The "B" address controller bits determine the type of comparison to be made:

15.6 SEARCH LINK LIST (SLL)/OP=37 (Continued)

B Address Controller		Search Type
00 (UN)	=	Search Equal. Set the Comparison Flags to EQUAL when the entire key field is equal to the "B" data field.
01 (SN)	=	Any Bit Equal. Set the Comparison Flags to EQUAL when any "one" bit of the key field is equal to the corresponding bit of the "B" data field.
10 (UA)	=	Less Than or Equal To. Set the Comparison Flags to LOW if the key field is algebraically less than the "B" data field. Set the Comparison Flags to EQUAL if the key field is equal to the "B" data field.
11 (IA)	=	No Bit Equal. Logical sums of corresponding bits of the "A" and "B" fields are compared. The logical sum is formed for each pair (all "B" field bits are examined). If the logical sums are zero (bit pairs 00, 01, or 10) for all pairs, set the Comparison Flags to EQUAL.

15.6 SEARCH LINK LIST (SLL)/OP=37 (Continued)

Function

The Search Link List instruction compares the key with the data located at "B+BF" or "B+2BF", as specified by the "A" address controller. If the comparison condition is met, set the Comparison Flags, as indicated above, and store the resolved "B" address, with its associated Base Indicant digit, in index register one (IX1). The standard EBCDIC sign is stored in sign digit of IX1. See Appendix A - Compatibility Notes (A.26.1).

The first six digits of "B" contain the address of the next list entry. In UA format, the address is in the first three characters.

If the conditions are not met, read the next list entry from the "B" data field. This list entry is a link to a new data field which replaces the original "B" data field address, however, the "B" address Base Indicant remains the same as for the resolved "B" data field address. BF is used in the same manner as it was with the original "B" data field. Cotinue this prccess until the list entry address is zero; at that time set the Comparison Flags to HIGH and terminate the instruction without storing into the index register (IX1).

The final "B" address will be checked for undigits (new link-list address will also be checked for undigits).

15.6 SEARCH LINK LIST (SLL)/OP=37 (Continued)

Examples

EXAMPLE (1) Search Equal

```
OP AF BF       A              B
-- -- --      ---            ---
37 05 06, A FIELD (UN), B-FIELD (UN)
```

	BEFORE	AFTER
A FIELD	12345	unchanged
B FIELD	00400012345	unchanged
IX1	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

EXAMPLE (2) Search Any Bit Equal, None Found

```
OP AF BF       A              B
-- -- --      ---            ---
37 01 06  A FIELD (UN), B-FIELD (SN)
```

	BEFORE	AFTER
A FIELD	6	unchanged
B FIELD	0000009	unchanged
IX1	nnnnnnnn	unchanged
COMPARISON	nnn	HIGH

15.6 SEARCH LINK LIST (SLL)/OP=37 (Continued)

EXAMPLE (3) Search Less than or Equal To

```
OP AF BF        A              B
-- -- --       ---            ---
37 05 06, A FIELD (UN), B-FIELD (UA)
```

	BEFORE	AFTER
A FIELD	12345	unchanged
B FIELD	00400012345	unchanged
IX1	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

EXAMPLE (4) Search No Bit Equal

```
OP AF BF        A              B
-- -- --       ---            ---
37 01 06  A-FIELD (UN), B-FIELD (IA)
```

	BEFORE	AFTER
A FIELD	6	unchanged
B FIELD	0000009	unchanged
IX1	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

15.7 SEARCH LINK DELINK (SLD)/OP=38

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 38

AF = Length of the "A" data field. May be indirect or literal flag. A value of "00" is equal to a length of 100 units.

BF = Amount of offset in units from the "B" address to the field to be searched. BF is typically six digits or more to allow for the link address at "B". A value of "00" is equal to an offset of zero units. BF may be indirect.

A = Address of the key to which the "B" data field will be compared. Address may be indexed, indirect or extended. The final address controller specifies the format for both the "A" and "B" fields and must be UN or UA. If SN is specified, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.26.2).

B = Address of the first list entry. The initial address may be indexed or extended. Indirect addressing is not allowed. The data format is that of the final "A" address controller. The "B" address controller bits determine the type of comparison to be made:

15.7 SEARCH LINK DELINK (SLD)/OP=38 (Continued)

B Address Controller		Search Type
00 (UN)	=	Search Equal. Set the Comparison Flags to EQUAL when the entire "A" key field is equal to the "B" data field.
01 (SN)	=	Any Bit Equal. Set the Comparison Flags to EQUAL when any "one" bit of the "A" key field is equal to the corresponding bit of the "B" data field.
10 (UA)	=	Less Than or Equal To. Set the Comparison Flags to LOW if the "A" key field is algebraically less than the "B" data field. Set the Comparison Flags to EQUAL if the "A" key field is equal to the "B" data field.
11 (IA)	=	No Bit Equal. Logical sums of corresponding bits of the "A" and "B" fields are compared. The logical sum is formed for each pair (all "B" field bits are examined). If the logical sums are zero (bit pairs 00, 01, or 10) for all pairs, set the Comparison Flags to EQUAL.

15.7 SEARCH LINK DELINK (SLD)/OP=38 (Continued)

Function

The Search Link Delink instruction compares the key with the data located at "B+BF" or "B+2BF", as specified by the "A" address controller. If the comparison condition is met, set the Comparison Flags as indicated above. If the condition is met the first time, store the resolved "B" address, with its associated Base Indicant, in both index register one (IX1) and two (IX2). On any other time, store the current "B" address into index register one (IX1) and store the previous "B" address into index register two (IX2). The standard EBCDIC sign is stored in sign digit of IX1 and IX2. See Appendix A - Compatibility Notes (A.26.1).

The first six digits of "B" contain the address of the next list entry. In UA format, the address is in the first three characters.

If the conditions are not met, read the next list entry from the "B" data field. This list entry is a link to a new data field which replaces the original "B" data field address, however, the Base Indicant remains the same as the original "B" data field address. BF is used in the same manner as it was with the original "B" data field. Continue this procedure until the list entry address is zero; at that time set the Comparison Flags to HIGH and terminate the instruction without storing into the index registers (IX1 & IX2).

The final "B" address will be checked for undigits (new link-list address will also be checked for undigits).

15.7 SEARCH LINK DELINK (SLD)/OP=38 (Continued)

Examples

EXAMPLE (1) Search Equal

```
OP AF BF        A               B
-- -- --       ---             ---
38 05 06  A FIELD (UN)   B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	12345	unchanged
B FIELD	00400012345	unchanged
004000	00500012345	00500012345
IX1	nnnnnnnn	B-FIELD ADDRESS
IX2	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

EXAMPLE (2) Search Any Bit Equal, None Found

```
OP AF BF        A               B
-- -- --       ---             ---
38 01 06  A FIELD (UN)   B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	6	unchanged
B FIELD	0000009	unchanged
IX1	nnnnnnnn	unchanged
IX2	nnnnnnnn	unchanged
COMPARISON	nnn	HIGH

15.7 SEARCH LINK DELINK (SLD)/OP=38 (Continued)

EXAMPLE (3) Search Less Than or Equal To

```
OP AF BF       A                B
-- -- --      ---              ---
38 05 06  A FIELD (UN)    B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	12345	unchanged
B FIELD	00400002345	unchanged
004000	00500012345	unchanged
IX1	nnnnnnnn	B-FIELD ADDRESS
IX2	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

EXAMPLE (4) Search No Bit Equal

```
OP AF BF       A                B
-- -- --      ---              ---
38 01 06  A FIELD (UN)    B FIELD (IA)
```

	BEFORE	AFTER
A FIELD	6	unchanged
B FIELD	0040006	unchanged
004000	0050009	unchanged
IX1	nnnnnnnn	00004000
IX2	nnnnnnnn	B-FIELD ADDRESS
COMPARISON	nnn	EQUAL

15.8 SEARCH LIST (SLT)/OP=64

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 64

AF = Unused and reserved. AF may be specified as an indirect field length but a literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Search Variant and may be specified as an indirect field length. The following variants may be specified by this field after any Indirect Field Length has been resolved:

FUNCTION	BF MSD
Store IX2 (Delink)	4
Normal - IX1 only	0

COMPARISON	BF LSD
Search Lowest	9
Search Highest	8
No Bit Equal	7
Any Bit Equal	6
A Greater Than or Equal to B	5
A Greater Than B	4
A Less Than or Equal to B	3
A Less Than B	2
A Not Equal to B	1
A Equal to B	0

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

A = Address of the key field. Address may be indexed, indirect or extended. The final address controller specifies the data type for both the key (A) and the comparison. The address controller must specify UN or UA. An SN controller will cause an Invalid Instruction fault (IEX = 03).

B = Address of the list field entry pointer. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03). This six digit field contains an address that is relative to the same memory area as the "B" address. This address is a pointer to the first list to be compared. A value of "EEEEEE" indicates an empty or "NULL" list.

C = Address of the list descriptor. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03). The length of this field is always 18 digits and in the following format:

INFORMATION		DIGITS
Link Offset	(digits)	00-05
Comparison Offset	(digits)	06-11
Key Length	(digits)	12-17

Note - The lowest memory address = 00

If any of the list descriptor values are invalid (undigits), cause an Address Error fault (AEX = 34).

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

The Search List instruction is a general search instruction for Linked Lists.

If the initial value of the list field entry pointer (B) is equal to "EEEEEE", the list is empty. Store the value "CiEEEEEE", where "i" represents the Base Indicant of the resolved "B" operand, in IX1 and set the Comparison Flags to NULL. If the store of IX2 is specified by the most significant digit of "BF", store the address of the List Field Entry pointer (B), relative to the same base as the resolved "B" operand, in IX2.

If the list is not empty, compare the data contained in the key (A), with a length as specified by the Key Length (C 12:6), with the data located in a specified list field. The "B" address specifies a location in memory that contains the six digit address, relative to the same memory area as the resolved "B" operand, of the list field entry pointer in memory. The list field key is found by adding the comparison offset (C 06:6) to the value of the list field entry pointer. The result of the comparison will cause one of two actions.

1. Except in the case of Search Lowest or Search Highest, if the comparison condition, as specified by the least significant digit of "BF" is met, store the list field entry pointer, relative to the same base as the "B" operand, in IX1.

   In the case of Search Lowest or Search Highest, the entire list is examined before storing the address of the entry with the lowest or highest value in IX1.

   If the store of IX2 is specified by the most significant digit of "BF" and this is the first comparison, store the address of the List Field Entry pointer, relative to the same base as the resolved "B" operand, in IX2. If it is other than the first comparison, store the address of the previous link address field (list field entry pointer plus the link offset), relative to the same base as the resolved "B" operand, in IX2.

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

2. If the selected comparison condition is not met, use the sum of the list field entry pointer and the Link Offset (C 00:6) as an address to obtain the six digit link address of the next field entry pointer from memory. Repeat this procedure until the compare condition is met or the link address is equal to "EEEEEE".

   If the link address is equal to "EEEEEE", store the NULL list value (CiEEEEEE) in IX1. If the store of IX2 is specified by the most significant digit of "BF", store the address of the link address field (list field entry point plus the link offset) in the last entry in the list, relative to the same base as the resolved "B" address, in IX2.

ANY BIT EQUAL

"Any Bit Equal" requires that all of the key (A) field be ANDed with all of the comparison field to determine if the the result is equal to zero. If the result is not equal to zero, a match occured.

NO BIT EQUAL

"No Bit Equal" requires that all of the key (A) field be ANDed with all of the comparison field to determine if the result is equal to zero. If the result is equal to zero, a match occured.

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

SEARCH LOWEST

Terminate a "Search Lowest" only when a NULL Link is reached. Search through the list for the lowest comparison field entry which is also less than key (A). If at least one entry is found, then store the address of the first such entry in IX1 with the same base indicant as the resolved "B" address.

If the store of IX2 is specified by the most significant digit of "BF", store the address of the previous link address field (list field entry point plus the link offset), relative to the same base as the resolved "B" address, in IX2.

If NO entries are found that are less than the key (A), store the NULL list value (CiEEEEEE) in IX1. If the store of IX2 is specified by the most significant digit of "BF", store the address of the link address field (list field entry point plus the link offset) in the last entry of the list, relative to the same base as the resolved "B" address, in IX2.

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

SEARCH HIGHEST

Terminate a "Search Highest" only when a NULL Link is reached. Search through the list for the highest comparison field entry which is also higher than key (A). If at least one entry is found, then store the address of the first such entry in IX1 with the same base indicant as the resolved "B" address.

If the store of IX2 is specified by the most significant digit of "BF", store the address of the previous link address field (list field entry point plus the link offset), relative to the same base as the resolved "B" address, in IX2.

If NO entries are found that are greater than the key (A), store the NULL list value (CiEEEEEE) in IX1. If the store of IX2 is specified by the most significant digit of "BF", store the address of the link address field (list field entry point plus the link offset) in the last entry of the list, relative to the same base as the resolved "B" address, in IX2.

The relative addresses stored in IX1 and IX2 will be relative to the same base as the resolved "B" address and contain the Base Indicant associated with the resolved "B" address.

The list must reside within one memory area as specified by the Base Indicant associated with the resolved "B" address. The processor will not check for improper memory assignments.

15.8 SEARCH LIST (SLT)/OP=64 (Continued)

Comparison Flags

If the comparison condition is met on the first entry, set the Comparison Flags to LOW. If the comparison condition is met on other than the first entry, set the Comparison Flags to EQUAL. If the comparison condition is not met, set the Comparison Flags to HIGH.

If the List is empty, set the Comparison Flags to NULL.

15.9 SEARCH TABLE (STB)/OP=66

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 66

AF = Unused and reserved. AF may be specified as an indirect field length but a literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Search Variant and may be specified as an indirect field length. The following variants may be specified by this field after any Indirect Field Length has been resolved:

COMPARISON	BF
Search Lowest	09
Search Highest	08
No Bit Equal	07
Any Bit Equal	06
A Greater Than or Equal to B	05
A Greater Than B	04
A Less Than or Equal to B	03
A Less Than B	02
A Not Equal to B	01
A Equal to B	00

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

15.9 SEARCH TABLE (STB)/OP=66 (Continued)

A = Address of the key field. Address may be indexed, indirect or extended. The final address controller specifies the data type for both the key (A) and the comparison. The address controller must specify UN or UA. An SN controller will cause an Invalid Instruction fault (IEX = 03).

B = Address of the base of the table. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

C = Address of the table descriptor. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03). The length of this field is always 24 digits and in the following format:

INFORMATION		DIGITS
Table Entry Length	(digits)	00-05
Comparison Offset	(digits)	06-11
Key Length	(digits)	12-17
Table Limit	(address)	18-23

Note - The lowest memory address = 00

Invalid table descriptor values will cause an Invalid Instruction fault (IEX = 07).

15.9 SEARCH TABLE (STB)/OP=66 (Continued)

The Search Table instruction is a general search instruction for Tables. The data contained in the key (A) is compared with the data located in a specified table field. Except for Search Lowest or Search Highest, If the comparison condition, as specified by the least significant digit of "BF" is met, store the address of the table entry in IX1 and set the Comparison Flags. If the selected condition is not met, examine the next table entry. Continue this procedure until the next table entry address is equal to or exceeds the Table Limit address.

In the case of Search Lowest or Search Highest, the entire table is examined before storing the address of the entry with the lowest or highest value in IX1.

Add the value of the Comparison Offset (C 06:6) to the Table Base address (B) to find the first field to be compared.

If the address of the first field to be compared is equal or greater than the Table Limit, the table is empty. Store the value "CiEEEEEE", where "i" represents the Base Indicant of the resolved "B" address, in IX1 and set the Comparison Flags to NULL.

If the comparison condition is not met, the sum of the Table Base address (B) and the Table Entry Length (C 00:6) replaces the Table Base address to point at the next table entry. If the next table entry address is equal to or exceeds the Table Limit address (C 18:6), set the Comparison Flags to HIGH and terminate the instruction. Otherwise, execute another comparison using data from the new table entry.

15.9 SEARCH TABLE (STB)/OP=66 (Continued)

Except in the case of Search Lowest or Search Highest, if the Table Limit is reached or exceeded, set the Comparison Flags to HIGH and terminate the instruction as indicated above, with IX1 containing "CiEEEEEE".

The relative address stored in IX1 contains the Base Indicant associated with the resolved "B" address.

The table must reside within one memory area as specified by the Base Indicant associated with the resolved "B" address. The processor will not check for improper memory assignments.

ANY BIT EQUAL

"Any Bit Equal" requires that all of the key (A) field be logically ANDed with all of the comparison field to determine if the the result is equal to zero. If the result is not equal to zero, a match occured.

NO BIT EQUAL

"No Bit Equal" requires that all of the key (A) field be logically ANDed with all of the comparison field to determine if the result is equal to zero. If the result is equal to zero, a match occured.

SEARCH LOWEST

Terminate a "Search Lowest" only when the Table Limit has been exceeded. If any comparison field entries are found that are less than the key (A), store the Table Entry pointer, relative to the same base as the resolved "B" address, for the FIRST FIELD WHICH IS LESS THAN OR EQUAL TO ALL THOSE FIELDS THAT ARE LESS THAN THE KEY, in IX1 with the same Base Indicant as the resolved "B" address.

If NO entries are found that are less than the key (A), store the NULL list value (CiEEEEEE) in IX1.

15.9 SEARCH TABLE (STB)/OP=66 (Continued)

SEARCH HIGHEST

Terminate a "Search Highest" only when the Table Limit has been exceeded. If any comparison field entries are found that are greater than the key (A), store the Table Entry pointer, relative to the same base as the resolved "B" address, for the FIRST FIELD WHICH IS GREATER THAN OR EQUAL TO ALL THOSE FIELDS THAT ARE GREATER THAN THE KEY, in IX1 with the same Base Indicant as the resolved "B" address.

If NO entries are found that are greater than the key (A), store the NULL list value (CiEEEEEE) in IX1.

Comparison Flags

If the comparison condition is met on the first compare, set the Comparison Flags to LOW. If the comparison condition is met on other than the first compare, set the Comparison Flags to EQUAL. If the comparison condition is not met, set the Comparison Flags to HIGH.

If the Table is empty, set the Comparison Flags to NULL.

15.10 BIT ZERO TEST (BZT)/OP=40

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 40

AF = Length of the "A" data field. May be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Eight bit selection mask. "One" bits in this mask select those bit positions to be tested for "zero" bits within each eight bit group of the "A" data field. "A" through "F" may be used to specify undigits in the mask. The field will not be recognized as indirect.

A = Address of the data field to be examined. Address may be indexed, indirect or extended. The final address controller type must be UN or UA. Use of SN data type will cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.05).

Function

The Bit Zero Test instruction tests a data field in memory (A) in eight-bit groups, for "zero" bits in the bit positions selected by the field mask (BF).

If the number of digits accessed is even, the entire eight-bit mask is applied to successive groups of two digits. If the number of digits is odd, the operation is the same until the last digit is accessed. The most significant four bits of the mask are applied to this digit.

15.10 BIT ZERO TEST (BZT)/OP=40 (Continued)

Comparison Flags

Set the Comparison Flags to EQUAL if any tested bit is "zero" or to HIGH if all tested bits are "one" bits.

Overlap

Field overlap is not applicable to this instruction.

Examples

EXAMPLE (1) Zero Test-Zero Found

```
OP AF BF       A
-- -- --      ---
40 04 C0, A FIELD (UA)

                DATA                      BINARY VALUE

A FIELD       C3C1E77B         11000011110000011110011101111011

MASK          C0C0C0C0         11000000110000001100000011000000
HIT              1                                     1
COMPARISON      EQUAL
```

EXAMPLE (2) Zero Test-All Ones Found

```
OP AF BF       A
-- -- --      ---
40 04 C0, A FIELD (UA)

                DATA                      BINARY VALUE

A FIELD       C2D9C1C3         11000010110110011100000111000011

MASK          C0C0C0C0         11000000110000001100000011000000
HIT            (no hit)
COMPARISON      HIGH
```

15.11 BIT ONE TEST (BOT)/OP=41

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 41

AF = Length of the "A" data field. May be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Eight bit selection mask. "One" bits in this mask select those bit positions to be tested for "one" bits within each eight bit group of the "A" data field. "A" through "F" may be used to specify undigits in the mask. The field will not be recognized as indirect.

A = Address of the data field to be examined. Address may be indexed, indirect or extended. The final address controller data type must be UN or UA. Use of SN data type will cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.05).

Function

The Bit One Test instruction tests a data field in memory (A) in eight-bit groups, for "one" bits in the bit positions selected by the field mask (BF).

If the number of digits accessed is even, the entire eight-bit mask is applied to successive groups of two digits. If the number of digits is odd, the operation is the same until the last digit is accessed. The most significant four bits of the mask are applied to this digit.

15.11 BIT ONE TEST (BOT)/OP=41 (Continued)

Comparison Flags

Set the Comparison Flags to EQUAL if any tested bit is a "one" or to HIGH if all tested bits are "zero" bits.

Overlap

Field overlap is not applicable to this instruction.

EXAMPLES

EXAMPLE (1) Ones Test-One Found

```
OP AF BF       A
-- -- --      ---
41 03 F0, A FIELD (UN)

                      DATA          BINARY VALUE

    A FIELD            001        0000 0000 0001
    MASK               F0F        1111 0000 1111
    HIT                  1                     1

    COMPARISON        EQUAL
```

EXAMPLE (2) Ones Test-All Zeros Found

```
OP AF BF       A
-- -- --      ---
41 02 03, A FIELD (UA)

                      DATA         BINARY VALUE

    A FIELD           C4C4      1100010011000100
    MASK              0303      0000001100000011
    HIT             (no hit)

    COMPARISON        HIGH
```

15.12 COMPARE ALPHA (CPA)/OP=45

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 45

AF = Length of "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or indicate the A-syllable is a literal.

BF = Length of "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect .

A = Address of the "A" data field. Address may be indexed, indirect or extended. The final address controller data type must be UA or UN and the same as the "B" address controller data type or cause an Invalid Instruction fault (IEX = 03).

B = Address of the "B" data field. Address may be indexed, indirect or extended. The final address controller data type must be UA or UN and the same as the "A" address controller data type or cause an Invalid Instruction fault (IEX = 03).

Note: Use of SN data types or mixed UA and UN data types will cause an Invalid Instruction fault (IEX = 03). See Appendix A, Compatibility Notes (A.02).

Function

The Compare Alpha instruction compares the characters (or digits) in the two data fields in memory according to the binary collating sequence, and sets the Comparison Flags accordingly.

15.12 COMPARE ALPHA (CPA)/OP=45 (Continued)

If the field lengths are unequal, and the data types are UA, pad the shorter field with trailing blanks (EBCDIC Code 40) to equal the length of the longer field. If the data types are UN, pad the shorter field with trailing zeros.

The values in memory are unchanged.

Comparison Flags

Set the Comparison Flags to HIGH if the binary value of the "A" data field is greater than that of the "B" data field, EQUAL if the two data fields have exactly the same bit patterns (including trailing blanks or zeros), and LOW if the binary value of the "A" data field is less than that of the "B" data field.

Overlap

There are no field overlap restrictions for this instruction.

15.12 COMPARE ALPHA (CPA)/OP=45 (Continued)

Examples

EXAMPLE (1) Compare Two Alpha Data Fields

```
OP AF BF       A             B
-- -- --      ---           ---
45 05 03, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1E3E24040	unchanged
B FIELD	C1E3E2	unchanged
COMPARISON	nnn	EQUAL

EXAMPLE (2) Compare Two Alpha Data Fields

```
OP AF BF       A             B
-- -- --      ---           ---
45 02 02, A FIELD (UA), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	C1D5	unchanged
B FIELD	C2D5	unchanged
COMPARISON	nnn	LOW

15.13 COMPARE NUMERIC (CPN)/OP=46

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 46

AF = Length of "A" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "A" address controller). AF may be indirect or may indicate the A-syllable is a literal.

BF = Length of "B" field. A value of "00" is equal to a length of 100 units (digits or characters as specified by the "B" address controller). BF may be indirect.

A = Address of the "A" data field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA.

B = Address of the "B" data field. Address may be indexed, indirect or extended. The final address controller data type may be UN, SN, or UA.

Function

The Compare Numeric instruction algebraically compares the numeric portion of the "A" data field in memory against the numeric portion of the "B" data field in memory.

The numeric portion of a UA data field consists of the least significant digit of each character (i.e. zone digits are ignored).

If the field lengths are unequal, pad the shorter field with leading zeros to equal the length of the longer field. The length does not include the sign digit of a signed numeric (SN) field.

15.13 COMPARE NUMERIC (CPN)/OP=46 (Continued)

Plus zero compares equal to minus zero.

UA and UN fields are assumed to have a positive sign.

The values in memory are unchanged.

Comparison Flags

Set the Comparison Flags to HIGH if the algebraic value of the numeric portion of the "A" data field is greater than that of the numeric portion of the "B" data field, EQUAL if the numeric portion of both data fields have exactly the same bit patterns (including leading zeros), and LOW if the algebraic value of the numeric portion of the "A" data field is less than that of the numeric portion of the "B" data field.

Overlap

There are no field overlap restrictions for this instruction.

15.13 COMPARE NUMERIC (CPN)/OP=46 (Continued)

Examples

EXAMPLE (1) Compare a Signed Literal Field with an Unsigned Field

```
OP AF BF    A                B
-- -- --   ---              ---
46 AA 05 C20 (SL)      B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	C20	unchanged
B FIELD	00015	unchanged
COMPARISON	nnn	HIGH

EXAMPLE (2) Compare a Numeric Literal Field with a Signed Field

```
OP AF BF     A                B
-- -- --    ---              ---
46 A6 02, 000012 (NL), B FIELD (SN)
```

	BEFORE	AFTER
A FIELD	000012	unchanged
B FIELD	C25	unchanged
COMPARISON	nnn	LOW

15.13 COMPARE NUMERIC (CPN)/OP=46 (Continued)

EXAMPLE (3) Compare a Numeric Field with an Alpha Field

```
OP AF BF       A              B
-- -- --      ---            ---
46 03 03, A FIELD (UN), B FIELD (UA)
```

	BEFORE	AFTER
A FIELD	213	unchanged
B FIELD	D2C1D4	unchanged
COMPARISON	nnn	LOW

15.14 BIT RESET (BRT)/OP=33

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 33

AF = Length of the "A" data field. May be indirect. A value of "00" is equal to a length of 100 units. A literal will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A.22).

BF = Eight bit selection mask. "One" bits in this mask select those bit positions to be set to "zero" bits within each eight bit group of the "A" data field. "A" through "F" may be used to specify undigits in the mask. The field will not be recognized as indirect.

A = Address of the data field operand. Address may be indexed, indirect or extended. The final address controller must be UN or UA. If UN format is specified and the number of digits accessed is even, the entire eight-bit mask is applied to successive groups of two digits. If the number of digits is odd, the operation is the same until the last digit is accessed. The most significant four bits of the mask are applied to this digit. If the controller specifies SN, cause an Invalid Instruction fault (IEX = 03).

Function

The Bit Reset instruction resets bits in a data field in memory (A) in eight-bit groups, according to the bit positions selected by "one" bits in the field mask (BF).

15.14 BIT RESET (BRT)/OP=33 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the least significant bit of the result is a "one"; otherwise, set them to EQUAL.

Overlap

There are no overlap restrictions.

Examples

EXAMPLE (1) Bit Reset, Alpha Field

```
OP AF BF       A
-- -- --      ---
33 03 A0, A FIELD (UA)
```

	DATA	BINARY VALUE
A FIELD	F1F2F3	111100011111001011110011
MASK	A0A0A0	101000001010000010100000
RESULT	515253	010100010101001001010011

COMPARISON HIGH

EXAMPLE (2) Bit Reset, Numeric Field

```
OP AF BF       A
-- -- --      ---
33 05 15, A FIELD (UN)
```

	DATA	BINARY VALUE
A FIELD	43105	01000011000100000101
MASK	15151	00010101000101010001
RESULT	42004	01000010000000000100

COMPARISON EQUAL

15.15 BIT SET (BST)/OP=34

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 34

AF = Length of the "A" data field. May be indirect. A value of "00" is equal to a length of 100 units. A literal will cause an Invalid Instruction fault (IEX = 21). See Appendix A - Compatibility Notes (A. 26.1).

BF = Eight bit selection mask. "One" bits in this mask select those bit positions to be set to "one" bits within each eight bit group of the "A" data field. "A" through "F" may be used to specify undigits in the mask. The field will not be recognized as indirect.

A = Address of the data field operand. Address may be indexed, indirect or extended. The final address controller must be UN or UA. If UN format is specified and the number of digits accessed is even, the entire eight-bit mask is applied to successive groups of two digits. If the number of digits is odd, the operation is the same until the last digit is accessed. The most significant four bits of the mask are applied to this digit. If the controller specifies SN, cause an Invalid Instruction fault (IEX = 03).

Function

The Bit Set instruction sets bits in a data field in memory (A), in eight-bit groups, according to the bit positions selected by "one" bits in the field mask (BF).

15.15 BIT SET (BST)/OP=34 (Continued)

Comparison Flags

Set the Comparison Flags to HIGH if the least significant bit of the result is a "one"; otherwise, set them to EQUAL.

Overlap

There are no overlap restrictions.

Examples

EXAMPLE (1) Bit Set, Alpha Field

```
OP AF BF        A
-- -- --       ---
34 03 A0, A FIELD (UA)
```

	DATA	BINARY VALUE
A FIELD	515253	010100010101001001010011
MASK	A0A0A0	101000001010000010100000
RESULT	F1F2F3	111100011111001011110011
COMPARISON	HIGH	

EXAMPLE (2) Bit Set, Numeric Field

```
OP AF BF        A
-- -- --       ---
34 05 F1, A FIELD (UN)
```

	DATA	BINARY VALUE
A FIELD	94236	10010100001000110110
MASK	F1F1F	11110001111100011111
RESULT	F5F3F	11110101111100111111
COMPARISON	HIGH	

15.16 LOGICAL AND (AND)/OP=42

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 42

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the "A" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

B = Address of the "B" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

C = Address of the result field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

Note: If the data types are not all UA or not all UN, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.11.1).

15.16 LOGICAL AND (AND)/OP=42 (Continued)

Function

The Logical And instruction stores the logical product (AND) of two data fields (A & B), located in memory, into a third memory location (C).

The "C" field length is equal to the larger of AF or BF. If the "A" and "B" fields are not of equal length, pad the shorter by adding trailing characters/digits of all zero bits.

Comparison Flags

Set the Comparison Flags to HIGH if the least significant bit of the result is a "one"; otherwise, set them to EQUAL.

Overlap

Total overlap or matching type-address overlap of any of the fields is allowed. Partial overlap of "A" or "B" with "C" other than matching type-address overlap may produce incompatible results. See Appendix A - Compatibility Notes (A.11.2).

15.16 LOGICAL AND (AND)/OP=42 (Continued)

Examples

EXAMPLE (1) AND Two Numeric Fields

```
OP AF BF       A            B            C
-- -- --      ---          ---          ---
42 02 03, A FIELD (UN), B FIELD (UN), C FIELD (UN)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	F6	unchanged	111101100000
B FIELD	235	unchanged	001000110101
C FIELD	nnn	220	001000100000
COMPARISON	nnn	EQUAL	

EXAMPLE (2) AND Two Alpha Fields

```
OP AF BF       A            B            C
-- -- --      ---          ---          ---
42 02 03, A FIELD (UA), B FIELD (UA), C FIELD (UA)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	E7E8	unchanged	111001111110100000000000
B FIELD	D4D8D1	unchanged	110101001101100011010001
C FIELD	nnn	C4C800	110001001100100000000000
COMPARISON	nnn	EQUAL	

15.17 LOGICAL OR (ORR)/OP=43

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 43

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the "A" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

B = Address of the "B" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

C = Address of the result field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

Note: If the data types are not all UA or not all UN, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.11.1).

15.17 LOGICAL OR (ORR)/OP=43 (Continued)

Function

The Logical Or instruction stores the logical sum (OR) of two data fields (A & B), located in memory, into a third memory location (C).

The "C" field length is equal to the larger of AF or BF. If the "A" and "B" fields are not of equal length, pad the shorter by adding trailing characters/digits of all "zero" bits.

Comparison Flags

Set the Comparison Flags to HIGH if the least significant bit of the result is a "one"; otherwise, set them to EQUAL.

Overlap

Total overlap or matching type-address overlap of any of the fields is allowed. Partial overlap of "A" or "B" with "C" other than matching type-address may produce incompatible results. See Appendix A - Compatibility Notes (A.11.2).

15.17 LOGICAL OR (ORR)/OP=43 (Continued)

Examples

EXAMPLE (1) OR Two Numeric Fields

```
OP AF BF       A                 B                 C
-- -- --      ---               ---               ---
43 02 03, A FIELD (UN), B FIELD (UN), C FIELD (UN)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	81	unchanged	100000010000
B FIELD	223	unchanged	001000100011
C FIELD	nnn	A33	101000110011
COMPARISON	nnn	HIGH	

EXAMPLE (2) OR Two Alpha Fields

```
OP AF BF       A                 B                 C
-- -- --      ---               ---               ---
43 03 02, A FIELD (UA), B FIELD (UA), C FIELD (UA)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	C1C2C4	unchanged	110000011100001011000100
B FIELD	F2F3	unchanged	111100101111001100000000
C FIELD	nnn	F3F3C4	111100111111001111000100
COMPARISON	nnn	EQUAL	

15.18 LOGICAL NOT (NOT)/OP=44

Format

```
+----+----+----+---------+---------+---------+
| OP | AF | BF |    A    |    B    |    C    |
+----+----+----+---------+---------+---------+
```

OP = 44

AF = Length of the "A" field. AF may be indirect or may indicate the A-syllable is a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the "B" field. BF may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the "A" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

B = Address of the "B" source field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

C = Address of the result field. Address may be indexed, indirect or extended. The final address controller data type may be UN or UA and must be the same as the other address controller data types.

Note: If the data types are not all UA or not all UN, cause an Invalid Instruction fault (IEX = 03). See Appendix A - Compatibility Notes (A.11.1).

15.18 LOGICAL NOT (NOT)/OP=44 (Continued)

Function

The Logical Not instruction stores the modulo two sum (Exclusive OR) of two data fields (A & B), located in memory, into a third memory location (C).

The "C" field length is equal to the larger of AF or BF. If the "A" and "B" fields are not of equal length, pad the shorter by adding trailing characters/digits of all "one" bits.

Comparison Flags

Set the Comparison Flags to set HIGH if the least significant bit of the result is a "one"; otherwise, set them to EQUAL.

Overlap

Total overlap or matching type-address overlap of any of the fields is allowed. Partial overlap of "A" or "B" with "C" other than matching type-address overlap may produce incompatible results. See Appendix A - Compatibility Notes (A.11.2).

15.18 LOGICAL NOT (NOT)/OP=44 (Continued)

<u>Examples</u>

EXAMPLE (1) Exclusive OR of Two Numeric Fields Function

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
44 03 03, A FIELD (UN), B FIELD (UN), C FIELD (UN)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	FFF	unchanged	111111111111
B FIELD	6A1	unchanged	011010100001
C FIELD	nnn	95E	100101011110
COMPARISON	nnn	EQUAL	

EXAMPLE (2) Exclusive OR of Two Alpha Fields

```
OP AF BF       A              B              C
-- -- --      ---            ---            ---
44 02 02, A FIELD (UA), B FIELD (UA), C FIELD (UA)
```

	BEFORE	AFTER	BINARY VALUE
A FIELD	5050	unchanged	0101000001010000
B FIELD	C7D7	unchanged	1100011111010111
C FIELD	nnnn	9787	1001011110000111
COMPARISON	nnn	HIGH	

16 INPUT/OUTPUT

16.1 INITIATE I/O (IIO)/OP=94

Format

```
+----+----+----+----------+
| OP | AF | BF |    A     |
+----+----+----+----------+
```

OP = 94

AF = AF may be specified as indirect or as a valid literal if the I/O descriptor does not require a length of greater than six digits. The length specified by AF is unused.

BF = Channel number. BF may be indirect.

A = Address of I/O descriptor. Address may be indexed, indirect or extended. The final address controller is ignored. The address must be modulo two.

Function

The Initiate I/O instruction causes the I/O Sub-system to receive an I/O Descriptor at a memory location (A) and route it to the appropriate I/O channel. See Appendix A - Compatibility Notes (A.24). If the I/O channel is busy, set the Comparison Flags to HIGH and terminate the instruction with no further action. Otherwise, set the Comparison Flags to EQUAL.

The format of the I/O descriptor is as follows:

INFORMATION	DIGITS
Opcode Syllable	00-05
A Address	06-15
B Address	16-25
C Field	26-33
Memory Area Status Number	34-39

This instruction may only be executed in Privileged Mode.

16.2 READ ADDRESS (RAD)/OP=92

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 92

AF = Operation variant. AF may be indirect, but a literal flag will cause an Invalid Instruction fault (IEX = 21). The following variants may be specified after any indirect field length has been resolved.

VARIANT	OPERATION
09	Store the 10 UN contents of memory specified by the "A" address into both the begin and the end address of the channel specified by BF.
03	Store up to 4 words (16 digits) of the extended R/D for the channel specified by BF in memory at the location specified by the "A" address. These words will be left-justified in a sixteen digit data field.
02	Store the first two words of the extended R/D for the channel specified by BF in memory at the location specified by the "A" address.
01	Store the current end address of the channel specified by BF in memory as a 10 UN field at the location specified by the "A" address.
00	Store the current begin address of the channel specified by BF in memory as a 10 UN field at the location specified by the "A" address.

The use of all other AF values is reserved and will cause an Invalid Instruction fault (IEX = 25).

16.2 READ ADDRESS (RAD)/OP=92 (Continued)

BF = Channel number. BF can be indirect. BF can specify any octal value from "00" to "77", or the non octal value "08". Use of other channel numbers will cause an Invalid Instruction fault (IEX = 26).

A = Address of the memory operand. The address may be indexed, indirect or extended. The final address controller is ignored. The address must be modulo two or cause an Address Error fault (AEX = 03).

Function

When BF equals a legal octal channel number from "00" to "77", the Read Address instruction causes the processor to read the specified data from an I/O channel address memory and store the value in a memory location (A) or write the value from a memory location (A) into both the begin and end addresses of the associated channel. See Appendix A - Compatibility Notes (A.24).

Examine the specified I/O channel to determine if it is busy with another I/O function. If it is available, the RAD function will be performed and the Comparison Flags set to EQUAL. If the I/O channel is unavailable, the RAD function will not be performed and the Comparison Flags will be set to HIGH.

A RAD to Channel 8 is treated as a no-op.

This instruction may only be executed in Privileged Mode.

16.3 SCAN RESULT DESCRIPTOR (SRD)/OP=91

Format

```
+----+--------+
| OP |  AFBF  |
+----+--------+
```

OP = 91

AFBF = Low order four digits of an absolute address in memory. The high order three digits are assumed to be zero. Indirect Field Lengths may be specified.

Function

If the resolved "AFBF" is equal to zero, set the Comparison Flags to EQUAL, and terminate the instruction without changing index register one (IX1).

If the resolved "AFBF" is non-zero, then the address formed from the "AFBF" is assumed to point to a sixteen bit result descriptor area. Examine the first bit of this area.

1. If it is equal to zero (no result descriptor present), examine the four digits (link address) immediately following the descriptor area.

   A. If they are equal to "0000", set the Comparison Flags to EQUAL and terminate the instruction (no descriptor found).

   B. If they are not equal to zero, they replace the original address value and the operation is repeated. The link addresses are assumed to be absolute addresses.

2. If it is equal to one (result descriptor present), store a sign and Base Indicant character of "C7" and six digits of the absolute address of the descriptor area into index register one (IX1).

16.3 SCAN RESULT DESCRIPTOR (SRD)/OP=91 (Continued)

Examine the next bit. Set the Comparison Flags to HIGH if it is a zero; otherwise, set them to LOW and terminate the instruction.

Undigits in the resolved AFBF or in the link addresses will cause an Address Error fault (AEX = 42). See Appendix A - Compatibility Notes (A.23)

This instruction may only be executed in Privileged Mode.

Comparison Flags

See functional description for details.

Overlap

There are no overlap cases for this instruction.

16.4 CONVERT I/O (CIO)/OP=85

Format

```
+----+----+----+---------+--------+
| OP | AF | BF |    A    |   B    |
+----+----+----+---------+--------+
```

OP = 85

AF = Unused and reserved. May be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Unused & reserved, but may be specified as an indirect field length.

A = Address of the initial descriptor. Address may be indexed indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the resultant descriptor. Address may be indexed indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

The format of the initial descriptor is as follows:

INFORMATION	DIGITS
Opcode Syllable	00-05
Environment Number	06-11
Memory Area Number	12-13
"A" Address (Convert)	14-19
"B" Address (Convert)	20-25
"C" Field	26-33

Note - The lowest memory address = 00

16.4 CONVERT I/O (CIO)/OP=85 (Continued)

The format of the resultant descriptor is as follows:

INFORMATION	DIGITS
Opcode Syllable	00-05
A Address	06-15
B Address	16-25
C Field	26-33
Memory Area Status Number	34-39

Function

This instruction converts the relative addresses in the initial descriptor to absolute addresses in the resultant descriptor, verifies that an I/O can be initiated to the specified memory area, and increments the "Number of I/O's in Process" field for that memory area in the Memory Area Status Table.

The initial descriptor (A) is read from memory. If the I/O descriptor "A" and "B" addresses are not mod 2 or contain undigits, or if the "A" address is not less than the "B" address, cause an Address Error fault (AEX = 01) and terminate the instruction with no further action.

Locate and resolve the Memory Area Table (MAT) entry pointed to by the Environment Number and Memory Area Number in the initial descriptor (A). If the resolved MAT entry is a Memory Area Fault Entry, then cause a Hard Memory Area Fault and terminate the instruction. Otherwise, if the resolved MAT entry is not an Original Entry, then cause an Address Error Fault (AEX = 04) and terminate the instruction. Add the base value in the resolved MAT entry to the initial descriptor "A" and "B" addresses and store them in the resultant descriptor "A" and "B" addresses. If the addresses are greater than the associated limit value, cause an Address Error fault (AEX = 24) and terminate the instruction with no further action.

16.4 CONVERT I/O (CIO)/OP=85 (Continued)

Move the initial descriptor Opcode Syllable and "C" field to the resultant descriptor. Move the Memory Area Status Table (MAST) Number from the resolved MAT entry to the resultant descriptor. Then use the MAST number as an array subscript into the Memory Area Status Table to locate the MAST entry associated with this memory area. Examine the Status digit of this Memory Area Status Table entry. If the Inhibited I/O Memory Area flag is set, set the Comparison Flags to LOW and terminate the instruction with no further action.

If the Inhibited I/O Memory Area flag is reset, increment the "Number of I/O's in Process" field in the Memory Area Status Table entry. If the result overflows the field, cause an Invalid Instruction fault (IEX = 05) and terminate the instruction with no further action. If no overflow, store the incremented number back in the field and set the comparison flags to EQUAL.

This instruction may only be executed in Privileged Mode.

16.5 I/O COMPLETE (IOC)/OP=98

Format

```
+----+----+----+---------+--------+
| OP | AF | BF |    A    |   B    |
+----+----+----+---------+--------+
```

OP = 98

AF = A length of six (6) must be specifiec directly or as an indirect field length or a literal.

BF = Channel number. BF may have an indirect field length. BF can specify any octal value from "00" to "77" or the non-octal value "08". Use of any other channel numbers will cause an Invalid Instruction fault (IEX = 26).

A = Address of the six digit Memory Area Status Table Number. Address may be indexed, indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the eight digit resultant field. Address may be indexed indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

16.5 I/O COMPLETE (IOC)/OP=98 (Continued)

Function

The I/O Complete instruction stores the unsigned difference between the I/O buffer begin and end address registers for the specified channel (BF) in memory (B). (The begin address will have been incremented during the I/O operation to show the number of bytes transferred.) This instruction examines the specified Memory Area Status Table entry (A), decrements the "I/Os in Process" field and sets the comparison flags accordingly.

1. If the I/O Channel is busy, set the Comparison Flags to HIGH and terminate the instruction with no further action.

   If the I/O Channel is not busy, continue as follows. If the specified channel number is the octal value 00 to 77, using the specified channel number, store the difference between the end address and the begin address at the specified memory location (B) If the specified channel number is the non-octal value 08, store zero at the specified memory location (B).

2. Use the Memory Area Status Table Number (A) as an array subscript into the Memory Area Status Table to locate the specified entry (See Section 5.6). If the Memory Area Status Table Number (A) is invalid (undigits), cause an Address Error fault (AEX = 34) and terminate the instruction with no further action.

3. Examine the "Number of I/O's in Process" field located in the Memory Area Status Table entry. If it is equal to zero, cause an Invalid Instruction fault (IEX = 05) and terminate the instruction with no further action. Otherwise, decrement the value of this field by one and store the result in the "Number of I/O's in Process" field of the specified MAST entry.

16.5 I/O COMPLETE (IOC)/OP=98 (Continued)

4. If the value of the "Number of I/O's in Process" field is now equal to zero, examine the Status digit of the Memory Area Status Table entry. If the Inhibited I/O Memory Area flag is set, set the Comparison Flags to LOW and terminate the instruction.

   If the Inhibited I/O Memory Area flag is reset or if the value of the "Number of I/O's in Process" field is not equal to zero, set the Comparison Flags to EQUAL and terminate the instruction.

This instruction may only be executed in Privileged Mode.

17 BINARY/DECIMAL CONVERSION

17.1 DECIMAL TO BINARY (D2B)/OP=88

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 88

AF = Length of the source data field. Value may be indirect or a literal. A length of of "00" is equal to a length of 100.

BF = Length of the destination data field in units consisting of four binary bits each. Value may be indirect. A value of "00" is equal to a length of 100 units.

A = Address of the decimal source data field. Address may be indirect, indexed or extended. The final address controller may be UN or UA. When the final controller is UA, the zone digits will be ignored. The final address controller must specify UN or UA or cause an Invalid Instruction fault (IEX = 03). Undigits in this field will cause an Invalid Arithmetic Data fault.

B = Address of the binary destination data field. Address may be indirect, indexed or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

17.1 DECIMAL TO BINARY (D2B)/OP=88 (Continued)

Function

The decimal to binary instruction will read a decimal data field from a memory location (A), convert the entire value to a binary representation and store the binary value in a second memory location (B).

If the converted data length is less than the destination data field (BF), store the converted data "Right Justified" with leading zeros.

If the converted value exceeds the length of the destination data field (BF), set the Overflow and terminate the instruction without storing the result.

Comparison Flags

Set the Comparison Flags to EQUAL if the source data field is equal to zero, otherwise, set them to "HIGH".

Overlap

This instruction has no overlap restrictions.

EXAMPLE: (1) Decimal to Binary

OP	AF	BF	A	B
88	03	02	A FIELD (UN)	B FIELD (UN)

	BEFORE	AFTER
A FIELD	174	unchanged
B FIELD	nn	AE
COMPARISON	nn	HIGH
OVERFLOW	nn	unchanged

17.1 DECIMAL TO BINARY (D2B)/OP=88 (Continued)

EXAMPLE: (2) Decimal to Binary - Overflow Condition

```
OP  AF  BF       A              B
--  --  --  ------------  ------------
88  03  02  A FIELD (UN)  B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	374	unchanged
B FIELD	nn	unchanged
COMPARISON	nnn	HIGH
OVERFLOW	nnn	ON

EXAMPLE: (3) Decimal to Binary - Mixed Controllers

```
OP  AF  BF       A              B
--  --  --  ------------  ------------
88  03  03  A FIELD (UA)  B FIELD (UN)
```

	BEFORE	AFTER
A FIELD	F1F7F4	unchanged
B FIELD	nnn	0AE
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

17.2 BINARY TO DECIMAL (B2D)/OP=89

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 89

AF = Length of the source data field in units of four binary bits each. Value may be indirect or a literal. A value of "00" is equal to a length of 100 units.

BF = Length of the destination data field. Value may be indirect. A value of "00" is equal to a length of 100.

A = Address of the binary source data field. Address may be indirect, indexed or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the decimal destination data field. Address may be indirect, indexed or extended. The final address controller may be UN or UA. When the final controller is UA, F zones will be inserted. The final address controller must specify UN or UA or cause an Invalid Instruction fault (IEX = 03).

Function

The binary to decimal instruction will read a binary data field from a memory location (A), convert the entire value to a decimal representation and store the decimal value in a second memory location (B).

If the converted data length is less than the destination data field (BF), store the converted data "Right Justified" with leading zeros.

17.2 BINARY TO DECIMAL (B2D)/OP=89 (Continued)

If the converted value exceeds the length of the destination data field (BF), set the Overflow Flag and terminate the instruction without storing the result.

Comparison Flags

Set the Comparison Flags to EQUAL if the source data field is equal to zero, otherwise, set them to "HIGH".

Overlap

This instruction has no overlap restrictions.

EXAMPLE: (1) Binary to Decimal

OP	AF	BF	A	B
89	02	03	A FIELD (UN)	B FIELD (UN)

	BEFORE	AFTER
A FIELD	AE	unchanged
B FIELD	nnn	174
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

17.2 BINARY TO DECIMAL (B2D)/OP=89 (Continued)

EXAMPLE: (2) Binary to Decimal - Overflow Condition

OP	AF	BF	A	B
88	02	02	A FIELD (UN)	B FIELD (UN)

	BEFORE	AFTER
A FIELD	AE	unchanged
B FIELD	nn	unchanged
OVERFLOW	nnn	ON
COMPARISON	nnn	HIGH

Example: (3) Binary to Decimal - Zero Source Data

OP	AF	BF	A	B
88	12	01	A FIELD (UN)	B FIELD (UN)

	BEFORE	AFTER
A FIELD	000000000000	unchanged
B FIELD	n	0
COMPARISON	nnn	EQUAL
OVERFLOW	nnn	unchanged

Example: (4) Binary to Decimal - Mixed Controllers

OP	AF	BF	A	B
89	03	03	A FIELD (UN)	B FIELD (UA)

	BEFORE	AFTER
A FIELD	0AE	unchanged
B FIELD	nnn	F1F7F4
COMPARISON	nnn	HIGH
OVERFLOW	nnn	unchanged

18 TIME-OF-DAY TIMER

These instructions concern the operation of the time-of-day timer. The time-of-day timer is a twenty digit counter that counts up at a one microsecond rate. See Appendix A - Compatibility Notes (A.36).

The time-of-day timer has the following format:

INFORMATION	DIGITS
Year	00 - 03
Month	04 - 05
Day	06 - 07
Reserved	08
Microseconds	09 - 19

Note - The lowest memory address = 00

The time-of-day timer is set (OP = 97) once as a privileged instruction. It may be read (OP = 95) by any user that requires the time-of-day.

At midnight, software will initialize the day, month, year, and microseconds via the STT instruction.

18.1 READ TIME of DAY (RDT)/OP=95

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 95

AF = Unused & reserved, but may be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Unused & reserved, but may be specified as an indirect field length.

A = Address of the memory location where the twenty digit timer value is to be stored. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

Function

The Read Time of Day instruction will store the twenty digit time-of-day timer at the memory location specified by the "A" address.

See Appendix A - Compatibility Notes (A.24).

Comparison Flags

Not changed.

Overlap

There are no overlap cases in this instruction.

18.2 SET TIME of DAY (STT)/OP=97

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 97

AF = Unused & reserved, but may be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Unused & reserved, but may be specified as an indirect field length.

A = Address of the memory location where the twenty digit timer value is stored. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

Function

The Set Time of Day instruction will load the time of day timer with the twenty digit value located in memory at "A". If the value is invalid (undigits), cause an Invalid Instruction fault (IXE = 07) and terminate the instruction.

This instruction may only be executed in Privileged Mode.

See Appendix A - Compatibility Notes (A.24).

Comparison Flags

Not changed.

Overlap

There are no overlap cases in this instruction.

19 MEASUREMENT

The Measurement register is an eight digit register with outputs that are made available as external outputs of the processor so that they may be monitored by various hardware monitoring devices.

The format of the Measurement register is:

INFORMATION	DIGITS
User	00-05
Name	06-07

Note - The lowest memory address = 00

The MOPOK signal, available externally, will be held to "zero" at any time that the Measurement register is being changed and held to a "one" at all other times.

19.1 MEASUREMENT OP (MOP)/OP=87

Format

```
+----+----+----+---------+--------+
| OP | AF | BF |    A    |   B    |
+----+----+----+---------+--------+
```

OP = 87

AF = A length of six (6) must be specified directly or as an indirect field length or a literal.

BF = A length of six (6) must be specified directly or as an indirect field length.

A = Address of the Setting field. Address may be indexed indirect or extended. The final address controller controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the Mask field. Address may be indexed indirect or extended. The final address controller controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

19.1 MEASUREMENT OP (MOP)/OP=87 (Continued)

Function

The Measurement instruction is used to load the User portion of the Measurement register. The Measurement register is an eight digit register with outputs that are made available as external outputs of the processor so that they may be monitored by various hardware monitoring devices.

The six digit Mask field will be read from a memory location (B) and used to determine which bits in the Measurement register are capable of being changed by the Setting field which is also located in memory (A).

Each bit in the Mask field that is equal to a "one" will permit the corresponding bit in the Measurement register User field to assume the state of the corresponding bit in the Setting Field (A).

Each bit in the Mask field that is equal to a "zero" will prevent changes to the corresponding bit in the Measurement register User field.

The MOPOK signal, available externally, will be held to "zero" at any time that the Measurement register is being changed and held to a "one" at all other times.

The Measurement register is changed by the Virtual Branch Reinstate (OP = 93), Branch Communicate (OP = 30), Hyper Call (OP = 62), and Return (OP = 63) instructions. It is also changed by the Interrupt and Hardware Call procedures.

20 MISCELLANEOUS

These instructions facilitate the implementation of the V-Series operating system.

20.1 ALTER TABLE ENTRY (ATE)/OP=86

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 86

AF = Unused & reserved, but may be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Variant, may be specified as an indirect field length.

BF = 00 marks the selected Memory Area Table entry as Unused.

BF = 01 performs a copy of a Memory Area Table entry.

BF = 02 alters the task's Environment Table entry.

BF = 03 alters the task's Memory Area Table entry.

BF = 04 signals the processor that a Memory Area Table entry was modified by a non-ATE instruction.

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address of the source operand. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

2C.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

When BF = 00 or BF = 04, this operand is unused. However, this address must still have valid address syllable attributes.

B = Address of the destination operand. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

When BF = 04, this operand is unused. However, this address must still have valid address syllable attributes.

2C.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

Function

BF = 00 WRITE UNUSED MEMORY AREA TABLE ENTRY

The Write Unused Memory Area Table entry variant is used to store an Unused Entry in the Memory Area Table entry specified by the Environment Number and the Memory Area Number found in the destination operand (B).

The eight digit destination operand contains the following information which is necessary to locate the desired Memory Area Table Entry.

INFORMATION	DIGITS
Environment Number	00-05
Memory Area Number	06-07

Note - The lowest memory address = 00

If the Destination Write Enable bit in the Environment Table entry associated with the destination operand is set, the destination entry is located using the Environment Number and the Memory Area Number found in the destination operand (B). Otherwise, cause an Invalid Instruction fault (IEX = 36) and terminate the instruction with no further action.

If the Type digit of the destination entry is an Original Entry or a Memory Area Fault Entry, cause an Invalid Instruction fault (IEX = 35) and terminate the instruction with no further action. Otherwise, write an Unused Entry in the specified destination.

Reload the current Memory Area Table using the Active Environment Number.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

BF = 01 COPY MEMORY AREA TABLE ENTRY

The Copy Memory Area Table entry variant is used to generate a copy of the source Memory Area Table entry specified by the Environment Number and the Memory Area Number found in the source operand (A) and store the copy in the destination entry location specified by the Envirorment Number and the Memory Area Number found in the destination operand (B).

Each eight digit operand contains the following information which is necessary to locate the desired Memory Area Table Entry.

INFORMATION	DIGITS
Environment Number	00-05
Memory Area Number	06-07

Note - The lowest memory adoress = 00

The following operations are performed by this variant:

1. Resolve the destination Memory Area Table entry specified by the Environment Number and the Memory Area Number contained in the destination operand (B). If the Type digit of the resolved destination entry is an Original entry or a Memory Area Fault entry, cause an Invalid Instruction fault (IEX = 35) and terminate the instruction with no further action.

2. Resolve the source MAT entry specified by the Environment Number and the Memory Area Number found in the source operand (A). If either the Source Copy Enable bit in the Environment Table entry associated with the source operand or the Destination Write Enable bit in the Environment Table entry associated with the destination operand is not set, cause an Invalid Instruction fault (IEX = 36) and terminate the instruction with no further action.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

3. If the resolved sourced MAT entry belongs to the current task, then store a "C" Copy descriptor which points to that resolved source MAT entry.

   If the resolved source MAT entry belongs to another task, then store an "E" Copy descriptor entry, which points to that resolved source MAT entry.

4. Reload the current Memory Area Table using the Active Environment Number.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

BF = 02 ALTER TASK'S ENVIRONMENT TABLE ENTRY

The Alter Task's Environment Table Entry variant is used to alter an existing Environment Table entry, specified by the destination operand (B), to be a copy of (identical to) the Environment Table entry specified by the source operand (A).

The source operand (A) format is:

INFORMATION	OFFSET
Environment Number	00 - 05
Task Number	06 - 09

The destination operand (B) format is:

INFORMATION	OFFSET
Environment Number	00 - 05

The destination Environment Number must be a decimal non-zero value. The source Environment Number must be a valid MCP or USER Environment Number. The source Task Number must be a decimal value.

The following operations are performed by this variant:

1. Locate the source Reinstate List entry by using the Task number, contained within the source operand (A), as an array subscript into the Reinstate List.

2. Locate the source Environment Table by using the Environment Table address contained within the source Reinstate List entry. Use the Environment Number contained within the source operand (A) as an array subscript into this Environment Table. If the source Environment Number is greater than the Number of Entries in Environment Table field, contained within the source task's Reinstate List Entry, cause an Address Error fault (AEX = 57) and terminate the instruction with no further action.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

3. Locate the destination entry using the Environment Number contained in the destination operand (B) as an array subscript into the appropriate Environment Table. If the first digit of the Environment Number is equal to "0 - 9", this six digit number represents an array subscript into the current User Environment Table of 000000 to 999999. If the Environment Number is equal to or greater than the value of the Number of Entries in the USER Environment Table field (located in the Reinstate List Entry for this Task), cause an Address Error fault (AEX = 57) and terminate the instruction with no further action.

   If the first digit of the Environment Table is not "0-9", cause Address Error (AEX = 52) and terminate the instruction with no further action.

4. Copy the contents of the source Environment Table entry to the destination Environment Table entry.

5. Reload the current Memory Area Table using the Active Environment Number.

For proper operation of this variant the following points must be followed. However, these points are not enforced by the hardware.

1. The destination EN must never be equal to the currently active EN.

2. The source MAT (pointed to by the EN/Task number pair) can never contain copy descriptors because these copy descriptors will not resolve correctly. Any pointers that may be affected by this instruction must be recalculated.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

BF = 03 COPY ALTERNATE TASK'S MEMORY AREA TABLE ENTRY

The Copy Alternate Task's Memory Area Table entry variant is used to generate a copy of the source Memory Area Table entry, specified by the source operand (A) and store the copy in the current task's destination operand (B). The format of the source and destination operands are defined below.

The source operands (A) format is as follows:

Information	Offset
Environment Number	00 - 05
Memory Area Number	06 - 07
Task Number	08 - 11

The destination operands (B) format is (The format is identical to BF = 01 variant of the ATE source and destination operands):

Information	Offset
Environment Number	00 - 05
Memory Area Number	06 - 07

The following operations are performed by this variant:

1. Locate and resolve the destination MAT entry using the Environment Number and the Memory Area Number contained in the destination operand (B).

2. If the Type digit of the destination entry is an Original Entry or a Memory Area Fault Entry, cause an Invalid Instruction fault (IEX = 35) and terminate the instruction with no further action.

20.1 ALTER TABLE ENTRY (ATE)/OP=86 (Continued)

3. Locate and resolve the source MAT entry using the Task Number, Environment Number, and Memory Area Number contained in source operand (A).

4. If either the Source Copy Enable bit in the Environment Table entry associated with the resolved source entry OR the Destination Write Enable bit in the Environment Table entry associated with the destination operand are NOT SET, cause an Invalid Instruction fault (IEX = 36) and terminate the instruction with no further action.

5. Store an "E" Copy Type Entry, containing the absolute address of the resolved source entry (A), into the destination entry (B).

6. Use the Active Environment Number to reload the current Memory Area Table from memory using the Active Environment Number.

BF = 04 NOTIFICATION OF MAT MODIFICATION BY NON-ATE INSTRUCTION

This variant performs no significant operation other than to notify the processor that a Memory Area Table Entry was changed by an instruction other than an ATE instruction.

It is necessary for those processors which cache Memory Area Base/Limit pairs to always be notified by some sort of ATE instruction when a MAT entry is modified.

This instruction may only be executed in Privileged Mode.

20.2 LOAD INDEX REGISTERS (LIX)/OP=67

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 67

AF = Unused. AF may be indirect but a literal flag will cause an Invalid Instruction fault (IEC = 21).

BF = Load Variant and may be specified as an indirect field length. The variants are specified after any Indirect Field Length has been resolved.

The least significant digit of the BF field specifies the insert of a Base Indicant value into the result as follows:

VARIANT	BFL
Base Indicant Value #7	7
Base Indicant Value #6	6
Base Indicant Value #5	5
Base Indicant Value #4	4
Base Indicant Value #3	3
Base Indicant Value #2	2
Base Indicant Value #1	1
No Base Indicant Value	0

The use of all other BFL values is reserved and will cause an Invalid Instruction fault (IEX = 26).

20.2 LOAD INDEX REGISTERS (LIX)/OP=67 (Continued)

The most significant digit of the BF field specifies the Index Register to be loaded as follows:

VARIANT	BFM
Load Index Register #7	7
Load Index Register #6	6
Load Index Register #5	5
Load Index Register #4	4
Load Index Register #3	3
Load Index Register #2	2
Load Index Register #1	1
Load Mobil Index Registers (4)	0

The use of all other BFM values is reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address of the Index Register field. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

20.2 LOAD INDEX REGISTERS (LIX)/OP=67 (Continued)

Function

The Load Index Register instruction provides the memory address (A) of the starting location of either an eight digit field that contains the value that is to be loaded into the specified Index Register (BFM) or the starting location of a 32-digit field that contains the values that are to be loaded into the four Mobile Index Registers (IX4, IX5, IX6 & IX7).

Each eight digits represents an index register of the following format:

INFORMATION	DIGITS
Sign	00
Base Indicant	01
Address	02-07

Note - The lowest memory address = 00

If the load variant (BFM) specifies the load of a single register and if a Base Indicant is specified (BFM = 0/), the value contained in the least significant digit of BF is inserted into the Base Indicant digit of the specified Index Register. See Appendix A - Compatibility Notes (A.32) for valid Base Indicant values.

The Mobile Index Registers (IX4, IX5, IX6, & IX7) are registers in the hardware (i.e., not located in main memory). When the remaining Index Registers (IX1, IX2, & IX3) are loaded, the associated memory location (8, 16, & 24 relative to Base #0) will be updated to the value found in the "A" operand.

20.3 STORE INDEX REGISTERS (SIX)/OP=68

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 68

AF = Reserved and unused. AF may be indirect but a literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Store Variant and may be specified as an indirect field length. The following variants may be specified by this field after any Indirect Field Length has been resolved:

VARIANT	BFM
Store Index Register #7	7
Store Index Register #6	6
Store Index Register #5	5
Store Index Register #4	4
Store Index Register #3	3
Store Index Register #2	2
Store Index Register #1	1
Store Mobil Index Registers (4)	0

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address of the Index Register field. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

20.3 STORE INDEX REGISTERS (SIX)/OP=68 (Continued)

Function

The Store Index Register instruction provides the memory address (A) of the starting location of either an eight digit field that will be used to store the specified (BFM) Index Register or the starting location of a 32-digit field that will be used to store the the four Mobil Index Registers (IX4, IX5, IX6 & IX7).

Each eight digits represents an index register of the following format:

INFORMATION	DIGITS
Sign	00
Base Indicant	01
Address	02-07

Note - The lowest memory address = 00

See Appendix A - Compatibility Notes (A.32) for valid Base Indicant values.

20.4 LOCK/UNLOCK (LOK)/OP=60

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 60

AF = Unused & reserved, but may be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Instruction Variant and may be indirect.

VARIANT	FUNCTION
09	Event Cause and Reset
08	Event Reset and Wait
07	Test Happened Status
06	Event Reset
05	Event Wait
04	Event Cause
02	Conditional Lock
01	Unconditional Lock
00	Unlock

All other BF values are reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address of the Lock/Event structure. Address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

If "BF" has a value of 00-02, "A" represents a Lock Structure.

If "BF" has a value of 04-09, "A" represents an Event Structure.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

The Lock Structure format is as follows:

INFORMATION	DIGITS
Lock Status Field *	00-01
Lock Owner Field	02-05
Lock Waiter Link Field	06-09
Lock Number Field	10-13
Lock Number Link Field	14-17
Reserved	18-19

Note - The lowest memory address = 00

The Event Structure format is as follows:

INFORMATION	DIGITS
Event Status Field *	00-01
Event State Field	02-05
Event Waiter Link Field	06-09
Event Designator Field	10-13
Event Count	14-19

Note - The lowest memory address = 00

* See Appendix A - Compatibility Notes (A.37)

If any of the Lock/Event Structure values are invalid (undigits), cause an Invalid Instruction (IEX = 07).

Function

This instruction examines the Lock/Event Structure (A) and, according to its value and the instruction variant (BF), will modify, if necessary, the Lock/Event Structure (A) and associated lock fields within the Reinstate List entry for the current task and other tasks owning or contending for the lock or event.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

The processor must determine if a lock is owned or available. If the Owner Field of the Lock Structure is equal to zero, the lock is available. If it is not equal to zero, the lock is owned.

The processor must determine if an event has happened. If the Event State Field is all "zeroes", the event has happened. If the Event State Field is all hexadecimal "F"s, the event has not happened. If the Event State Field contains any other value, cause an invalid instruction fault (IEX = 06) and terminate the instruction with no further action.

The machine dependent Lock Status Field of the Lock/Event Structure (A) may also be used to represent the status of the structure with one value representing owned and another representing available. This specification will always refer to the Owner Field of the Lock Structure (A) with the understanding that some systems may incorporate this feature. See Appendix A - Compatibility Notes (A.37).

This instruction may only be executed in Privileged Mode.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 00 - UNLOCK

This variant releases a Lock and, if any task is waiting for this lock, causes an interrupt to the MCP Kernel.

Read the Lock Structure (A) from memory. The value of the "Lock Owner Field" must equal the current Task Number located in absolute memory location 82 or cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

Store zeros into the "Lock Owner Field" of the Lock Structure (A) to indicate that this lock is now available.

Compare the "Lock Number Field" of the Lock Structure (A) with the "MCP Canonical Lock Number" field located in the Reinstate List Entry for the current task. If they are not equal, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the number fields are equal, store the contents of the "Lock Number Link Field" of the Lock Structure (A) into the "MCP Canonical Lock Number" field, located in the Reinstate List Entry for the current task.

Examine the "Lock Waiter Field" of the Lock Structure (A). If it is equal to zero, set the Comparison Flags to EQUAL and terminate the instruction.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

If the "Lock Waiter Link Field" is not equal to zero, execute the following procedure.

1. The Reinstate List pointer has been located with a Write Hardware Register (OP = 65:BF = 00) instruction. The four digit "Lock Waiter Field" of the Lock Structure (A) is used as an array subscript into this table to locate a new Reinstate List Entry. A sign and Base Indicant character of "C7" and six digits of the absolute address of this Reinstate List Entry are stored absolute memory location 24 - 31 (absolute IX3).

2. Store the Released Lock Flag (03) into the Instruction Interrupt Cause Descriptor in absolute memory locations 32 - 33.

3. Set the Comparison Flags to HIGH and cause an Interrupt procedure to be executed that stores the address of the next instruction to be executed in the Interrupt Frame.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 01 - UNCONDITIONAL LOCK

This variant competes for the lock specified by the Lock Structure (A) and, if the lock is owned, causes an interrupt to the MCP Kernel.

Read the Lock Structure (A) from memory.

Compare the "Lock Number Field" of the Lock Structure (A) with the "MCP Canonical Lock Number" field, located in the Reinstate List Entry for the current task. If the "Lock Number Field" is less than or equal to the "MCP Canonical Lock Number" field, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the Lock is available, store the current task number (4 digits), located in absolute memory location 82, into the "Lock Owner Field" and store zeros into the "Lock Waiter Link Field" of the Lock Structure (A) and execute the following procedure.

1. Copy the contents of the "MCP Canonical Lock Number" field, located in the current Reinstate List Entry, into the "Lock Number Link Field" of the Lock Structure (A).

2. Copy The contents of the "Lock Number Field" of the Lock Structure (A) into the "MCP Canonical Lock Number" field, located in the Reinstate List Entry for the current task.

3. Set the Comparison Flags to EQUAL and terminate the instruction.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

If the Lock is owned, execute the following procedure.

1. Copy the "Lock Owner Field" of the Lock Structure (A) into the "Task Number Owning" field located in the Reinstate List Entry for the current task.

2. Copy the "Lock Waiter Link Field" of the Lock Structure (A) into the "Next Task in List" field located in the Reinstate List Entry for the current task.

3. The four digit "Lock Owner Field" of the Lock Structure (A) is used as an array subscript into the Reinstate List to locate a new entry. A sign and Base Indicant character of "C7" and six digits of the absolute address of this Reinstate List Entry are stored absolute memory location 24 - 31 (absolute IX3).

4. Store the current Task number, located at absolute memory location 82, into the "Lock Waiter Link Field" of the Lock Structure (A).

5. Store the Waiting Lock flag (01) into the "State Indicator" field located in the Reinstate List Entry for the current task.

6. Store the Failed Lock Flag (01) into the Instruction Interrupt Cause Descriptor in absolute memory locations 32 - 33.

7. Ignore the Trace Mode Bit. Even if in trace mode, do not perform a Trace Hardware Call following this instruction.

8. Set the Comparison Flags to LOW and cause an Instruction Interrupt to the MCP Kernel that stores the current instruction address in the Interrupt Frame.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 02 - CONDITIONAL LOCK

This variant attempts to obtain the lock specified by the Lock Structure (A).

If the Lock is available, perform the following:

1. Compare the "Lock Number Field" of the Lock Structure (A) with the "MCP Canonical Lock Number" field located in the Reinstate List Entry for the current task. If the "Lock Number Field" is less than or equal to the "MCP Canonical Lock Number" field, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

2. If the "Lock Number Field" is greater than the "MCP Canonical Lock Number, store the current task number (4 digits), located in absolute memory location 82, into the "Lock Owner Field" and store zeros into the "Lock Waiter Link Field" of the Lock Structure (A).

3. Copy the contents of the "MCP Canonical Lock Number" field, located in the Reinstate List Entry for the current task, into the "Lock Number Link Field" of the Lock Structure (A).

4. Copy the contents of the "Lock Number Field" of the Lock Structure (A) into the "MCP Canonical Lock Number" field, located in the Reinstate List Entry for the current task.

5. Set the Comparison Flags to EQUAL and terminate the instruction.

If the Lock is owned:

1. If the "Lock Owner Field" equals the Task Number, set the Comparison Flags to LOW and terminate the instruction with no further action. If the owner is not the current task, set the Comparison Flags to HIGH and terminate the instruction with no further action.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 04 - EVENT CAUSE

This variant causes an Event and signals this fact to all tasks that are waiting for this event.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not ecual to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the Event is in the Happened state, increment the "Event Count Field" by one (if it was originally at the maximum value for the container, set it to zero), set the Comparison Flags to Equal and terminate the instruction with no further action.

If the Event is in the Not Happened state, increment the "Event Count Field" of the Event structure (A) by one (if it was originally at the maximum value, set it to zero, and set the Event State Field to the Happened State.

Examine the "Event Waiter Link Field" of the Event Structure (A). If it is equal to zero, set the Comparison Flags to EQUAL and terminate the instruction with no further action.

If the "Event Waiter Link Field" is not equal to zero, execute the following procedure.

1. The four digit "Event Waiter Link Field" of the Event Structure (A) is used as an array subscript into the Reinstate List to locate a new entry. A sign and base indicant character of "C7" and six digits of the absolute address of this Reinstate List Entry are stored at absolute memory locations 24 - 31 (absolute IX3).

2. Store the Released Event Flag (04) into the Instruction Interrupt Descriptor in absolute memory locations 32 - 33.

3. Set the Comparison Flags to HIGH and cause an Instruction Interrupt to the MCP Kernel that stores the address of the next instruction to be executed in the Interrupt Frame.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 05 - EVENT WAIT

This variant will cause the current task to wait (i.e. be suspended) until the specified event is caused, if it is currently in the Not Happened State.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the event is in the Happened State, set the Comparison Flags to EQUAL and terminate the instruction with no further action.

If the event is in the Not Happened State, execute the following procedure.

1. Copy the "Event Waiter Link Field" of the Event Structure (A) into the "Next Task in List" field located in the Reinstate List Entry for the current task.

2. Store the current Task number, located at absolute memory location 82, into the "Event Waiter Link Field" of the Event Structure (A).

3. Store the Waiting Event flag (02) into the "State Indicator" field located in the Reinstate List Entry for the current task.

4. Store a Failed Event flag (02) into the Instruction Interrupt Cause Descriptor in absolute memory locations 32 - 33.

5. Set the Comparison Flags to LOW and cause an Instruction Interrupt to the MCP Kernel that stores the next instruction address in the Interrupt Frame.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 06 - EVENT RESET

This variant resets the Happened State of the event to the Not Happened State.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the event is in the Not Happened State, set the Comparison Flags to HIGH and terminate the instruction with no further action.

If the event is in the Happened State, reset the "Event State Field" to the Not Happened State and store zeroes into the "Event Waiter Link Field" of the Event Structure (A).

Set the Comparison Flags to EQUAL and terminate the instruction.

BF = 07 - EVENT TEST HAPPENED STATUS

This variant tests whether or not an Event Happened.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

If the event is in the Not Happened State, set the Comparison Flags to HIGH and terminate the instruction with no further action.

If the event is in the Happened State, set the Comparison Flags to EQUAL and terminate the instruction with no further action.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 08 - EVENT RESET AND WAIT

This variant resets an Event Structure to the Not Happened state and forces the current task to wait (e.g., become suspended) unit the event has been caused.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with not further action.

Reset the Event State Field to the Not Happened State, then execute the following procedure.

1. Copy the "Event Waiter Link Field" of the Event Structure (A) into the "Next Task In List" field located in the Reinstate List Entry for the current task.

2. Store the Current Task Number located in the current Reinstate List Entry into the "Event Waiter Link Field" of the Event Structure (A).

3. Store the Waiting Event flag (02) into the "State Indicator" field located in the Reinstate List Entry for the current task.

4. Store the Failed Event flag (02) into the Instruction Interrupt Cause Descriptor in absolute memory locations 32 - 33.

5. Set the Comparison Flags to "Low" and cause an Instruction Interrupt to the MCP Kernel that stores the next instruction address in the Interrupt Frame.

20.4 LOCK/UNLOCK (LOK)/OP=60 (Continued)

BF = 09 - EVENT CAUSE AND RESET

This variant causes an Event, allowing any tasks which were waiting for the event to continue processing, and leaves the Event Structure in the Not Happened state.

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

Increment the "Event Count Field" by one. Reset the Event State Field to the Not Happened state.

Examine the "Event Waiter Link Field" of the Event Structure (A). If it is equal to zero, set the Comparison Flags to "Equal" and terminate the instruction with no further action.

If the "Event Waiter Link Field" is not equal to zero, execute the following procedure.

1. The four-digit "Event Waiter Link Field" of the Event Structure (A) is used as an array subscript into the Reinstate List to locate a new entry. A sign and base indicant character of "C7" and six digits of the absolute address of this Reinstate List Entry are stored at absolute memory locations 24 - 31 (absolute IX3).

2. Store the Released Event flag (04) into the Instruction Interrupt Descriptor in absolute memory locations 32 - 33.

3. Set the Comparison Flags to "High" and cause an Instruction Interrupt to the MCP Kernel that stores the address of the next instruction to be executed to be saved in the Interrupt Frame.

20.5 INITIALIZE LOCK/EVENT STRUCTURES (ILS)/OP=69

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 69

AF = Length of the A operand in digits. AF may be specified as indirect or as a literal. Value has to be 1, 4, or 6, otherwise an Invalid Instruction fault (IEX = 20) will be caused.

BF = Instruction Variant and may be indirect.

Variant	Function
02	Counted Wait
01	Create Lock
00	Create Event

All other BF values are reserved and will cause an Invalid Instruction fault (IEX = 26) if used.

A = Address of the initial state data for the Event/Lock being initialized or used. The address may be indexed, indirect, or extended. The final address controller must be UN or an Invalid Instruction fault will be caused (IEX = 03). If "BF" is 00, this is the address of a Boolean value which determines whether the Event Structure being created will initially be in the Happened state or in the Not Happened state. If "BF" is 01, this is the address of the canonical number for the created Lock Structure. If "BF" is 02, this is the address of the count value to be used to determine whether to Wait or not.

20.5 INITIALIZE LOCK/EVENT STRUCTURES (ILS)/OP=69 (Continued)

B = Address of the Event/Lock Structure being initialized. The address may be indexed, indirect, or extended. The final address controller must be UN or an Invalid Instruction fault will be caused (IEX = 03). If "BF" is 00 or 02, this is the address of an Event Structure. If "BF" is 01, this is the address of a Lock Structure.

Function

This instruction creates and initializes a Lock Structure or an Event Structure in memory, or performs a counted wait on event. This instruction may only be executed in privileged mode.

20.5 INITIALIZE LOCK/EVENT STRUCTURES (ILS)/OP=69 (Continued)

BF = 00 - CREATE EVENT

This variant creates an Event Structure in either the Happened state or the Not Happened State.

If the A Operand is non-zero (Boolean true), create an Event Structure at the B Address in the Happened state. If the A Operand is zero (Boolean false), create the Event Structure in the Not Happened state. All other fields in the event are cleared to zeroes and the comparison flags remain unchanged.

BF = 01 - CREATE LOCK

This variant creates a Lock Structure in the available state with a canonical Lock Number as specified by the A Operand.

If the A Operand is zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action. If it contains undigits, cause an Invalid Instruction fault (IEX = 07) and terminate the instruction with no further action.

Create a Lock Structure in the processor-dependent available state, with the "Lock Number Field" set to the value provided by the A Operand and all other fields cleared to zero.

The comparison flags remain unchanged.

20.5 INITIALIZE LOCK/EVENT STRUCTURES (ILS)/OP=69 (Continued)

BF = 02 - COUNTED WAIT

This variant uses the count provided by the A Operand as a guard to determine whether to perform the "Wait" function on the Event Structure described by the B Address.

Read the Event Structure (B) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action.

Read the 6-digit A Operand from memory. If the value of the A Operand is not equal to the "Event Count Field" in the Event Structure (B), set the comparison flags to equal and terminate the instruction with no further action.

If the value of the 6-digit A operand is equal to the "Event Count Field" in the Event Structure (B), examine the processor-dependent state of the Event. If it is in the Happened state, set the comparison flags to EQUAL and terminate the instruction with no further action. If it is in the Not Happened state, execute the following procedure.

1. Copy the "Event Waiter Link Field" of the Event Structure (A) into the "Next Task In List" field located in the Reinstate List Entry for the current task.

2. Store the Current Task Number located in the current Reinstate List Entry into the "Event Waiter Link Field" of the Event Structure (A).

3. Store the Waiting Event flag (02) into the "State Indicator" field located in the Reinstate List Entry for the current task.

4. Store a Failed Event flag (02) into the Instruction Interrupt Cause Descriptor in absolute memory locations 32 - 33.

5. Set the Comparison Flags to LOW and cause an Instruction Interrupt to the MCP Kernel that stores the next instruction address in the Interrupt Frame.

20.6 MOVE LOCK STRTUCTURES (MLS)/OP=6A

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = 6A

AF = Unused and Reserved, but may be specified as an Indirect Field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Instruction Variant and may be indirect.

Variant	Function
01	Move Lock Owner
00	Move Event Count

A = Address of the Lock Structure or Event Structure. Address may be indexed, indirect, or extended. The final address controller must be UN or an Invalid Instruction fault will be caused (IEX = 03).

B = Address of the receiving field for the information being moved from the Lock Structure/Event Structure. The address may be indexed, indirect, or extended. The final address controller must be UN or an Invalid Instruction fault will be caused (IEX = 03).

Function

This operator moves the specified piece of state from an Event Structure (BF = 00) or a Lock Structure (BF = 01) to an identically sized destination field. This instruction may only be executed in privileged mode.

20.6 MOVE LOCK STRTUCTURES (MLS)/OP=6A (Continued)

BF = 00 - MOVE EVENT COUNT

Read the Event Structure (A) from memory. If the "Event Designator Field" is not equal to zero, cause an Invalid Instruction falult (IEX = 06) and terminate the instruction with no further action.

If the "Event Count Field" contains undigits, cause an Invalid Instruction Fault (IEX = 07) and terminate the instruction with no further action.

Move the 6 digit "Event Count Field" from the Event Structure (A) to the 6 digit destination field specified by the B Address.

BF = 01 - MOVE LOCK OWNER

Read the Lock Structure (A) from memory. If the "Lock Number Field" is equal to zero, cause an Invalid Instruction fault (IEX = 06) and terminate the instruction with no further action. If the "Lock Number Field" contains undigits, cause an Invalid Instruction Fault (IEX = 07) and terminate the instruction with no further action.

If the "Lock Number Field" is not equal to zero or contains no undigits, move the 4 digit Lock Owner Field from the Lock Structure (A) to the 4 digit destination field specified by the B Address.

20.7 WRITE HARDWARE REGISTERS (WHR)/OP=65

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 65

AF = Unused & reserved, but may be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Variant and may be specified as an indirect field length. The following variations may be specified by this field after any Indirect Field Length has been resolved:

BF = 00 REINSTATE LIST ADDRESS

Use the "A" operand to locate the nine digit absolute memory address of the Reinstate List pointer. See Appendix A - Compatibility Notes (A.38). Recalculate any references based on the Reinstate List pointer. See Appendix A - Compatibility Notes (A.43).

BF = 01 SNAP PICTURE ADDRESS

Use the "A" operand to locate the nine digit absolute memory address of the Snap Picture. Snap Picture Enable will be set to "one".

2C.7 WRITE HARDWARE REGISTERS (WHR)/OP=65 (Continued)

BF = 02 MEMORY ERROR REPORT ADDRESS

Use the "A" operand to locate the nine digit absolute memory address of the Memory Error Report. Memory Error Report Enable will be set to a "one" and Memory Error Report Pending will be set to "zero". Different processors have different requirements for the Memory Error Report. See Appendix A - Compatibility Notes (A.31).

BF = 03 MEMORY AREA STATUS TABLE ADDRESS

Use the "A" operand to locate the nine digit absolute memory address of the Memory Area Status Table. Recalculate any references based on this pointer.

OTHER BF VALUES

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address, in memory, of the data field. The address may be indexed, indirect or extended. The final address controller must equal UN or cause an Invalid Instruction fault (IEX = 03).

If any of the address values contained in the "A" data field are invalid (undigits), cause an Address Error fault (AEX = 34).

Note that certain task state is loaded by the processor when executing this instruction. See Section 4.1 for more details.

This instruction may only be executed in Privileged Mode.

20.8 SET MODE (SMF)/OP=47

Format

```
+----+--------+
| OP |  AFBF  |
+----+--------+
```

OP = 47

AFBF = Unused and ignored.

Function

The Set Mode instruction performs no useful operation. See Appendix A - Compatibility Notes (A.24).

20.9 FAIL (BAD)/OP=AB

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = AB

AF = Unused and ignored. The literal and indirect field length flags will be ignored.

BF = Unused and ignored. The indirect field length flag will be ignored.

A = Unused and ignored. The indirect, extented and indexed flags will be ignored.

Function

The Fail instruction causes an intentional Invalid Instruction fault (IEX = 01). The instruction will not be examined for any invalid address constraints.

20.10 SYSTEM STATUS (SST)/OP=99

Format

```
+----+----+----+---------+
| OP | AF | BF |    A    |
+----+----+----+---------+
```

OP = 99

AF = Unused and reserved. May be specified as an indirect field length. A literal flag will cause an Invalid Instruction fault (IEX = 21).

BF = Status Variant. May be specified as an indirect field length.

VARIANT	FUNCTION
00	STATUS INDICATIORS
01	SYSTEM I/D

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

A = Address of the Status data field. Address may be indexed, indirect or extended. The final address controller must equal UA or cause an Invalid Instruction fault (IEX = 03).

Function

The System Status instruction stores the number of bytes of status, as specified below, into the specified memory location (A).

This instruction may only be executed in Privileged Mode.

20.10 SYSTEM STATUS (SST)/OP=99 (Continued)

BF = 00 SYSTEM STATUS

The System Status is stored in memory in the following format:

INFORMATION	BYTE	BIT
Reserved (0)	00	4-7
Memory Error Report Status	00	3
Reserved	00	2
Temperature Warning Status	00	1
Voltage Warning Status	00	0
Machine Dependent Data	01-99	ALL *

Note - The lowest memory address = 00

* See Appendix A - Compatibility Notes (A.40).

BF = 01 SYSTEM I/D

The System I/D is stored in memory in the following format:

INFORMATION	EBCDIC BYTES	
Processor Type	00-09	*
Specification Level	10-19	*
Shared System Number	20-21	
Multiple Processor Number	22-23	
Serial Number	24-33	*
Memory Size	34-49	
Firmware Level	50-97	*
Reserved (00 00)	98-99	

Note - The lowest memory address = 00

* See Appendix A - Compatibility Notes (A.40).

The Firmware Level field of the System I/D must use the character "FF" to indicate the end of data within the field.

20.10.1 STATUS INDICATORS

The Status Indicators are described in the following paragraphs.

MEMORY ERROR REPORT STATUS

This bit is set to indicate that a Memory Error Report has been stored in memory at a location that has been previously been set with a Write Hardware Register instruction (OP = 65:BF = 02).

VOLTAGE WARNING STATUS

This bit is set to indicate that the System input Voltage is less than a preset value. This condition does not cause a power-off cycle. This bit will be true as long as the warning condition exists.

TEMPERATURE WARNING STATUS

This bit is set to indicate that the System Temperature has exceeded a preset value. This condition does not cause a power-off cycle. This bit will be true as long as the warning condition exists.

21 STRING INSTRUCTIONS

String instructions operate on large data fields that are described by string descriptors that are found in memory at the locations specified by the "A" and "B" instruction addresses.

The format of the string descriptor is as follows:

INFORMATION		DIGITS
Reserved		00-01
Environment Number		02-07
Memory Area Number		08-09
String Begin Address	(SBA)	10-15
String End Address	(SEA)	16-21
Container Begin Address	(CBA)	22-27
Container End Address	(CEA)	28-33

Note: - the lowest memory address = 00

The "4" bit of the most significant digit of the instruction field length (AF/BF) contains the Memory Area Variant.

If the Memory Area Variant is not set, then the Environment Number and the Memory Area Number contained in the string descriptor must be resolved to point to the selected Memory Area Table entry.

If the Memory Area Variant is set, then the addresses contained within the string descriptor are relative to the base of the same Memory Area that is specified for the address of the string descriptor.

If the first digit of the Environment Number is equal to a "D" or if the Environment Number is equal to "0" and the processor is not in Privileged Mode, cause an Invalid Instruction fault (IEX = 32) and terminate the instruction with no further action. Otherwise, resolve the Memory Area Table entry pointed to by the Environment Number and Memory Area Number in the string descriptor.

21 STRING INSTRUCTIONS (Continued)

If the resolved MAT entry type digit indicates an Original entry, use the Base and Limit addresses contained in the entry to locate the string.

If the resolved MAT entry type digit indicates an Unused entry, cause an Address Error (AEX = 50) fault that will store the instruction address of the String instruction in the resultant Hardware Call procedure and terminate the instruction with no further action.

If the resolved MAT entry type digit indicates a Memory Area Fault entry, cause a Hard Memory Area Fault that will store the address of the next instruction to be executed in the resultant Hardware Call procedure. The Environment Number, Memory Area Number, and Task Number that point to this Memory Area Table entry will be stored as stack parameters in the resultant Hardware Call procedure.

This procedure is repeated for each of the operands.

STRING BEGIN ADDRESS	- The six digit address, relative to the specified memory area, of the first digit of the string.
STRING END ADDRESS	- The six digit address, relative to the specified memory area, of the first digit beyond the end of the string.
CONTAINER BEGIN ADDRESS	- The six digit address, relative to the specified memory area, of the area allocated to contain the specified string.
CONTAINER END ADDRESS	- The six digit address, relative to the specified memory area, of the first digit beyond the area allocated to contain the specified string.

21.1 MOVE STRING (MVS)/OP=A0

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = A0

AF = Source field variant. AF may be indirect but a literal flag will cause an Invalid Instruction fault (IEX = 21).

The least significant digit is the Update Variant. A value of "0" indicates that no Update should take place. A value of "1" indicates that an update of the source string begin address should take place. The use of all other AFL values is reserved and will cause an Invalid Instruction fault (IEX = 25).

The most significant digit is the Memory Area Variant. A value of "0" indicates that the Memory Area specified by the Environment Number and the Memory Area Number contained in the source string descriptor (A) is to be used for the addresses contained within the source string descriptor. A value of "4" indicates that the Memory Area specified for the source string descriptor (A) is to be used for the addresses contained within the source string descriptor. The use of all other AFM values is reserved and will cause an Invalid Instruction fault (IEX = 25).

21.1 MOVE STRING (MVS)/OP=AO (Continued)

BF = Destination field variant. BF may be indirect.

The two least significant bits of the most significant digit of BF contain the Substring Select variant and are coded to select a substring from the destination string. The possible selections are:

SUB-STRING RANGE	VALUE
Container Begin => Container End	3
Reserved	2
String End => Container End	1
String Begin => String End	0

The "4" bit of the most significant digit of BF is the Memory Area Variant. If the variant is equal to "0", the Memory Area specified by the Environment Number and the Memory Area Number contained in the destination string descriptor (B) is to be used for the addresses contained within the destination string descriptor. If the variant is equal to "1", the Memory Area specified for the destination string descriptor (B) is to be used for the addresses contained within the destination string descriptor.

The least significant digit of BF is the Update Variant. A value of "0" indicates that no update of the destination string begin address should take place. A value of "1" indicates that an update should take place.

The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

21.1 MOVE STRING (MVS)/OP=A0 (Continued)

A = Address of the source string descriptor. Address may be indexed, indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the destination string descriptor. Address may be indexed, indirect or extended. The final address controller specifies the padding variant as follows:

PADDING VARIANT	B-CONTROLLER
Pad with zero	0 (UN)
No padding	1 (SN)
Pad with blank (40)	2 (UA)

21.1 MOVE STRING (MVS)/OP=A0 (Continued)

Function

The Move String instruction will move the string begin to string end substring from a location specified by the source string descriptor (A) to a location specified by the destination string descriptor (B) and the string select variant (BF). If padding is required in the destination, it will be specified by the Padding variant (BC). Data will not be read from a string end address or written into the string end address or the container end address.

If the source substring begin address is greater than the source substring end address, cause an Address Error fault (AEX = 01) and terminate the instruction with no further action.

If the destination substring begin address is greater than the destination substring end address, cause an Address Error fault (AEX = 01) and terminate the instruction with no further action.

If the Update bit in "AF" is equal to a "zero", the source string descriptor will not be changed.

If the Update bit in "AF" is equal to a "one", the String Begin Address in the source string descriptor is set to point to one digit beyond the last digit moved.

If the Update bit in "BF" is equal to a "zero", the destination string descriptor will not be changed.

If the Update bit in "BF" is equal to a "one", the String End Address in the destination string descriptor is set to point to one digit beyond the last digit written.

If the Update bit in "BF" is equal to a "one" and the Sub-string Select bits are equal to "3" (CBA => CEA), the String Begin Address in the destination string descriptor is set to the same value as the Container Begin Address.

21.1 MOVE STRING (MVS)/OP=A0 (Continued)

EQUAL LENGTHS

If the source and destination lengths are equal the source substring will be moved to the destination substring and the Comparison Flags will be set to EQUAL.

SOURCE LONGER THAN DESTINATION

If the source length is longer than the destination length, then a length corresponding to the destination length will be moved from the left justified source substring to the destination substring and the Comparison Flags will be set to HIGH.

SOURCE SHORTER THAN DESTINATION

If the source length is shorter than the destination length, then a length corresponding to the source length will be moved from the source substring to the left justified destination substring. The "B" address controller bits will determine if padding is required and what type of padding is required.

If the "B" address controller is equal to "1", then no padding takes place and the Comparison Flags are set to LOW.

If the "B" address controller is equal to "0", then the remaining digits in the destination string are padded with zeros and the Comparison Flags are set to EQUAL.

If the "B" address controller is equal to "2", then the remaining digits in the destination string are padded with blanks (40) and the Comparison Flags are set to EQUAL. If there are an odd number of digits remaining in the destination string, cause an Invalid Instruction fault (IEX = 07) and terminate the instruction without updating the pointers.

21.1 MOVE STRING (MVS)/OP=A0 (Continued)

Null Strings

If the source string is a null (SBA=BEA), then the destination field will be filled with the padding character specified by the "B" address controller. If the destination string is a null, the Comparison Flags will be set HIGH and the instruction will terminate with no further action. If both the source string and the destination string are null, the Comparison Flags will be set to EQUAL and the instruction will terminate with no further action.

Overlap

String containers and descriptors that occupy any of the same memory locations will produce unspecified results that may vary from processor model to processor model.

Partial overlapping string containers will produce unspecified results that may vary from processor model to processor model.

Partial overlapping descriptors will produce unspecified results that may vary from processor model to processor model.

Total overlap of string containers is permitted.

Total overlap of descriptors (A = B) is permitted. The hardware is restricted to not update the source descriptor before the destination descriptor has been acquired.

Partial overlap of substrings will produce unspecified results unless the destination substring select bits equal "3" (CBA => CEA) and the descriptors totally overlap (A = B), in which case the string is normalized by moving the string to the left and filling the container with the padding characters specified by the "B" address controller.

21.2 COMPARE STRING (CPS)/OP=A1

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = A1

AF = A Memory Area Variant. AF may be indirect but a literal flag will cause an Invalid Instruction fault (IEX = 21). The most significant digit is the Memory Area Variant. A value of "0" indicates that the Memory Area specified by the Environment Number and the Memory Area Number contained in the first string descriptor (A) is to be used for the addresses contained within the first string descriptor. A value of "4" indicates that the Memory Area specified for the first string descriptor (A) is to be used for the addresses contained within the first string descriptor. The use of all other AF values is reserved and will cause an Invalid Instruction fault (IEX = 25).

BF = B Memory Area Variant. BF may be indirect. The most significant digit is the Memory Area Variant. A value of "0" indicates that the Memory Area specified by the Environment Number and the Memory Area Number contained in the second string descriptor (B) is to be used for the addresses contained within the second string descriptor. A value of "4" indicates that the Memory Area specified for the second string descriptor (B) is to be used for the addresses contained within the second string descriptor. The use of all other BF values is reserved and will cause an Invalid Instruction fault (IEX = 26).

21.2 COMPARE STRING (CPS)/OP=A1 (Continued)

A = Address of the first string descriptor. Address may be indexed, indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the second string descriptor. Address may be indexed, indirect or extended. The final address controller specifies the padding variant as follows:

PADDING VARIANT	B-CONTROLLER
Pad with zero	0 (UN)
No padding	1 (SN)
Pad with blank (40)	2 (UA)

Function

The Compare String instruction will compare the binary values of the substring defined by the String Begin and End Addresses specified by the "A" string descriptor to the substring defined by the String Begin and End Addresses specified by the "B" string descriptor and set the Comparison Flags EQUAL if the strings are identical, HIGH if the "A" string is of greater value than the "B" string, and LOW if the "A" string is of lesser value than the "B" string. No compare will take place on the data located at the string end address.

21.2 COMPARE STRING (CPS)/OP=A1 (Continued)

If a String Begin Address is greater than the String End Address, cause an Address Error fault (AEX = 01) and terminate the instruction with no further action.

The comparison of two null strings (strings of zero length) will cause the Comparison Flags to be set to EQUAL.

If the source and destination lengths are equal the strings will be compared and the Comparison Flags will be set.

If the lengths are unequal, the comparison will be dependent on the value of the "B" address controller.

If the "B" address controller is equal to "1", the longer string will be compared to the shorter string for the length of the shorter string only. However, for this case of no padding, comparison with a null string will always set the Comparison Flags EQUAL.

If the "B" address controller is equal to "0", the longer string will be compared to the shorter string for the length of the shorter string then the remainder of the longer string will be compared against zeros. (For comparison with a null string, the non-null string will be compared entirely against zeroes.)

If the "B" address controller is equal to "2", the longer string will be compared to the shorter string for the length of the shorter string then the remainder of the longer string will be compared against blank (40) characters. (For comparison with a null string, the non-null string will be compared entirely against blank characters.)

Overlap

Partial overlapping descriptors will produce unspecified results that may vary from processor model to processor model.

21.3 HASH STRING (HSH)/OP=A2

Format

```
+----+----+----+---------+---------+
| OP | AF | BF |    A    |    B    |
+----+----+----+---------+---------+
```

OP = A2

AF = A Memory Area Variant. AF may be indirect but a literal flag will cause an Invalid Instruction fault (IEX = 21). A value of "00" indicates that the Memory Area specified by the Environment Number and the Memory Area Number contained in the source string descriptor (A) is to be used for the addresses contained within the source string descriptor. A value of "40" indicates that the Memory Area specified for the source string descriptor (A) is to be used for the addresses contained within the source string descriptor. The use of all other AF values is reserved and will cause an Invalid Instruction fault (IEX = 25).

BF = Length of the "B" field. A value of "00" is equal to a length of 100 digits. BF may be indirect.

A = Address of the source string descriptor. Address may be indexed, indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

B = Address of the destination hash key field. Address may be indexed, indirect or extended. The final address controller must specify UN or cause an Invalid Instruction fault (IEX = 03).

21.3 HASH STRING (HSH)/OP=A2 (Continued)

Function

The Hash String instruction will produce a hash key of "BF" digits based on the string defined by the String Begin and End Addresses specified by the "A" string descriptor and store the key in the memory location specified by the "B" address.

If the String Begin Address is greater than the String End Address, cause an Address Error fault (AEX = 01) and terminate the instruction with no further action.

If the length of the source string is less than the length of the destination field (BF), the data from the source string will be moved to the destination data field and the remaining destination data field will be filled with trailing zeros.

If the length of the source string is equal to the length of the destination field (BF), the data from the source string will be moved to the destination data field and the instruction will terminate with no further action.

21.3 HASH STRING (HSH)/OP=A2 (Continued)

The hashing algorithm requires that successive "BF" amounts of the specified string are Exclusive OR'ed and the result stored in memory. The following steps illustrate one method of performing this algorithm.

1. Using the String Begin Address as the source address, move "BF" digits from the string to the "B" field in memory.

2. Increment the source address by "BF".

3. Perform an Exclusive OR of "BF" digits specified by the source address and the "B" field in memory.

4. Store the result in the "B" field in memory.

5. Increment the source address by "BF" and repeat Steps 3 - 5 until the difference between the source address and the String End Address is zero or less than "BF".

   If the difference is zero the instruction is complete.

   If the difference is less than "BF", then perform the next Exclusive OR with only the difference amount.

   Store the result in the "B" field in memory leaving the remainder of the "B" field as it was previously and terminate the instruction.

Overlap

Any overlap of the "B" operand with the descriptor or the string produces unspecified results.

22 RESERVED MEMORY

22.1 KERNEL DATA AREA

The following areas of absolute memory are reserved for the purposes indicated.

Absolute Memory Address	Purpose
00-39	Indirect Field Length
00-07	Undefined
08-15	Index Register One (IX1)
16-23	Index Register Two (IX2)
24-31	Index Register Three (IX3)
21-22	Interrupts Occured Code
32-33	Instruction Interrupt Cause Descriptor
34-35	MCP Kernel Request Code
38-39	SCAN Result Storage
40-45	Kernel Stack Pointer
46-47	Breakpoint Pattern for Kernel
48	HALT Execution Digit
49	Internal I/O Mask
50-71	Unused
72-81	R/D Storage Area
82-85	Current Task Number
86-93	Reserved
94-99	Kernel Interrupt Branch Address
100-111	Channel "00" Result Descriptor and Link
120-131	Channel "01" Result Descriptor and Link
.	.
.	.
.	.
1640-1651	Channel "77" Result Descriptor and Link
8000-8039	MCP Kernel Request Data

Figure 20-1 Channel Format

```
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|01|02|03|04|05|06|07|08|09|10|11|12|13|14|15|16|17|18|19|20|
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|           |           |           |                       |
|<---R/D--->|<---LINK-->|<---R/D--->|<------MCP USAGE------>|
```

Memory address for channel "nn" Result Descriptor and Link equals:

20 x Channel "nn" + 100.

22.2 RESERVED MEMORY RELATIVE TO THE MCP DATA AREA

Each task has the following areas of its MCP Data Area (Environment #0, Memory Area #0) reserved for the purpose indicated.

Relative Memory Address	Purpose
00-39	Indirect Field Length
08-15	Index Register One (IX1)
16-23	Index Register Two (IX2)
24-31	Index Register Three (IX3)
38-39	SCAN Result Storage
40-45	Stack Pointer
46-47	Breakpoint Bit Pattern
48-49	Edit Table Entry 0
50-51	Edit Table Entry 1
52-53	Edit Table Entry 2
54-55	Edit Table Entry 3
56-57	Edit Table Entry 4
58-59	Edit Table Entry 5
60-61	Edit Table Entry 6
62-63	Edit Table Entry 7
64-65	Trap Enable (FF)
66-71	Trap Address
72-81	R/D Storage Area
82-85	Task Number
86	Reserved
87-92	Hyper Call Function Table Pointer
93	Reserved
94-99	Hyper Call Function Table Limit

APPENDIX A - COMPATIBILITY NOTES

INTRODUCTION

The main body of this specification describes the machine independent behavior of a family of computers built with the V-Series operating system architecture. All members of the family exhibit the operational characteristics described in the main body. Previously supported user programs are still supported by this architecure, however, all of the prior privileged instructions have been changed or modified so that only a V-Series operating system may be executed with the describle instructions.

The purpose of this appendix is to describe and contrast the machine dependent behavior of various medium systems processors as applied to the specified instructions.

Note: In the descriptions that follow, the phrase "results are unpredictable" means that the instruction may not get the same results from execution to execution due to the parallel access bidding by multiple requesters, the fact that input operands may not be fully buffered before writing, and the requirement that in some cases multiple memory modules must be available for a read.

A.01 RELATED SPECIFICATIONS

V3 -

PS 1994 2382	V300	System
SDS 1987 1193	V300	Architecture
EDS 1983 6915	V300	IOP
EDS 1990 9431	V300	SNAP

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.02 CPA

"Use of SN data types or mixed UA, UN data types may produce incompatible results."

B4800 - Mixed UN, UA data types:

Each digit of UN field is appended to a zero digit to produce a character of the form 0d which is compared against the other operand's character.

SN data type:

The B4800 treats this exactly as if it were UN (eg. 7 SN is treated as if 7 UN).

B2900 - Both operand data types must be the same, or else
B3900 it is treated as an Invalid Instruction. SN is treated as UN.

B4900 - If the data types are not both UA or both UN, execution of the instruction results in a BCT to 94 with Invalid Instruction set in the processor R/D.

V3 - If the data types are not both UA or both UN, execution of the instruction results in a Hardware Call procedure with an Invalid Instruction fault (IEX = 03).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.03 MVA

"When the "A" and "B" controllers indicate UA data, the field lengths are equal (AF=BF), and the final "B" address is within the "A" data field (address "A" to "A" + 2 x AF), the source data field between the "A" and "B" address will be repeated throughout the destination field."

"Cases of overlapping "A" and "B", other than described above, may produce incompatible results."

B4800 - UN-UN or SN-SN:

Source data between "A" and "B" addresses will be replicated in "B". Rules for padding or truncation remain the same.

UA-UA:

AF does not have to equal BF for replication.

Mixed data types cause unusual results.

B2900 - In all cases, other than the one described, no
B3900 replication will take place. The result field will contain the same data as if there were no overlap.

B4900 - Same as B2900/B3900 except that partial
V3 overlapping the literal field will cause the result field to contain the same data as if there were no overlap.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.04 MVN

A.04.1 "Cases of overlapping "A" and "B", other than described above, may produce incompatible results."

B4800 - UN-UN, SN-SN or UA-UA:

Source data between "A" and "B" addresses will be replicated in "B". Rules for padding or truncation remain the same.

Mixed data types cause unusual results.

B2900 - In all cases, other than the one described, no
B3900 replication will take place. The result field
B4900 will contain the same data as if there were no
V3 overlap.

A.04.2 "Move Numeric UA-UA and UA-UN cause incompatible result in the final comparison flags."

B4800 - UA-UA, UA-UN
B4900
V3

If the interpreted value of the source data is zero, the comparison flags will be set to EQUAL, otherwise the comparison flags will be set to HIGH if the first digit of the source data is interpreted as positive or the comparison flags will be set to LOW if the source data is interpreted as negative.

B2900 - UA-UA, UA-UN
B3900

If the interpreted value of the source data is zero, the comparison flags will be set to EQUAL, otherwise the comparison flags will be set HIGH.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.05 BZT, BOT

"Use of SN data type may produce incompatible results.

B4800 - Treats all non-UA data types as UN.

B2900 - An SN data type will result in an Invalid
B3900   Instruction.
B4900

V3 - An SN data type will cause an Invalid Instruction fault (IEX = 03).

A.06 LSS, EQL, LEQ, GTR, NEQ, GEQ, NUL, GTN

"Use of branch prediction op codes may result in incompatible behavior."

B4900 - Employs a branch prediction scheme whereby
V3   various op codes indicate the most probable branch path.

APPENDIX A - COMPATIBILITY NOTES (Continued)

Branch Prediction Op	Equivalent Branch Op	Predicted Branch Path
21	21(LSS)	Not taken (last time not taken)
22	22(EQL)	Not taken (last time not taken)
23	23(LEQ)	Not taken (last time not taken)
24	24(GTR)	Not taken (last time not taken)
25	25(NEQ)	Not taken (last time not taken)
26	26(GEQ)	Not taken (last time not taken)
2A	2A(NUL)	Not taken (last time not taken)
2B	2B(GTN)	Not taken (last time not taken)
B1	21(LSS)	Not taken (last time taken)
B2	22(EQL)	Not taken (last time taken)
B3	23(LEQ)	Not taken (last time taken)
B4	24(GTR)	Not taken (last time taken)
B5	25(NEQ)	Not taken (last time taken)
B6	26(GEQ)	Not taken (last time taken)
BA	2A(NUL)	Not taken (last time taken)
BB	2B(GTN)	Not taken (last time taken)
E1	21(LSS)	Taken (last time not taken)
E2	22(EQL)	Taken (last time not taken)
E3	23(LEQ)	Taken (last time not taken)
E4	24(GTR)	Taken (last time not taken)
E5	25(NEQ)	Taken (last time not taken)
E6	26(GEQ)	Taken (last time not taken)
EA	2A(NUL)	Taken (last time not taken)
EB	2B(GTN)	Taken (last time not taken)
F1	21(LSS)	Taken (last time taken)
F2	22(EQL)	Taken (last time taken)
F3	23(LEQ)	Taken (last time taken)
F4	24(GTR)	Taken (last time taken)
F5	25(NEQ)	Taken (last time taken)
F6	26(GEQ)	Taken (last time taken)
FA	2A(NUL)	Taken (last time taken)
FB	2B(GTN)	Taken (last time taken)

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.07 MVW, MVC

A.07.1 "If AF indicates literal, the results may be incompatible."

B4800 - Since the field length is AFBF, the normal literal code (e.g. "A6") would indicate a literal of over 400 digits starting at the instruction's A-syllable.

B2900 - Results in an Address Error.
B3900

B4900 - Results in an Invalid Instruction.

V3 - Causes an Invalid Instruction fault (IEX = 21).

A.07.2 "Use of non-Mod 4 "A" or "B" addresses may produce incompatible results."

B4800 - Results in an Address Error.

B2900 - No mod restrictions on "A" or "B".
B3900
B4900
V3

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.08 INC, DEC

"Partial overlap of "A" or "B" may produce imcompatible results."

B4800 - Results are consistent but undefined.

B2900 - There are no overlap restrictions. The correct
B3900 result will be stored in the "B" field.

B4900 - If both the "A" and "B" operands are less than or
V3 equal to 10 digits long, the correct answer will be stored in the result field. If either "A" or "B" operands are greater than 10 digits long and if the "A" address equals the "B" address and their data types are also the same, the correct result will be stored regardless of the values for AF and BF. In all other cases, the results are undefined.

A.09 ADD, SUB

"Partial overlap of "A" or "B" with "C" may produce imcompatible results."

B4800 - Results are undefined.

B2900 - Since there are no overlap restrictions, the
B3900 correct result will be stored in the "C" field.

B4900 - If both the "A" and "B" operands are less than or
V3 equal to 10 digits long, the correct answer will be stored in the result field. If either "A" or "B" operands are greater than 10 digits long and if the "A" or "B" address equals the "C" address and the respective data types are also the same, the correct result will be stored regardless of the values for AF and BF. In all other cases, the results are undefined.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.10 MPY

"Partial overlap of "A" or "B" with "C" may produce incompatible results."

B4800 - Results are consistent but undefined.

B2900 - Since there are no overlap restrictions, the
B3900 correct result will be stored in the "C" field.

B4900 - If both the "A" and "B" operands are less than or
V3 equal to 10 digits long, the correct answer will be stored in the result field. If either "A" or "B" operands are greater than 10 digits long and if the "A" or "B" address equals the "C" address and the respective data types are also the same, the correct result will be stored regardless of the values for AF and BF. In all other cases, the results are undefined.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.11 AND, ORR, NOT

A.11.1 "If the data types are not all UA and not all UN, incompatible results may be produced."

B4800 - Mixed data types produce irregular results. For example, UN-UN-UA case results in characters whose zone digits are zero. In the UN-UA-UN case, the least significant digit of each "B" field character is ignored.

B2900 - The "A" data type is used for all three fields
B3900 ("B" and "C" data types are ignored).

B4900 - Results in an Invalid Instruction.

V3 - Cause an Invalid Instruction fault (IEX = 03).

A.11.2 "Partial overlap of "A" or "B" with "C" may produce incompatible results".

B4800 - Results are produced from previous intermediate results.

B2900 - There are no overlap restrictions. The correct
B3900 result will be stored in the "C" field.

B4900 - If both the "A" and "B" operands are less than
V3 or equal to 10 digits long, the correct answer will be stored in the result field. If either "A" or "B" operands are greater than 10 digits long and if the "A" or "B" address equals the "C" address and the respective data types are also the same, the correct result will be stored regardless of the values for AF and BF. In all other cases, the results are undefined.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.12 SEA

"In each type of search, if the "B" field entry being compared to the key overlaps with the "C" address location, incompatible results may be produced."

B4800 -	Search for Equal.
B4900	
V3	The overlapping "B" field entry is always considered Not equal to the key.
	Search for Low or Lowest.
	Only those units of the "B" field entry up to the "C" address are compared against a corresponding number of units of the "A" field key.
B2900 - B3900	If the "B" field overlaps the "C" address and it is the first comparison (i.e., starting "B" address), then the full "B" field is compared. If the "B" field overlaps the "C" address and it is not the first comparison, no comparison is performed and this "B" field entry is considered not equal to the key.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.13 EDT

A.13.1 "Use of an SN data type for the "C" field may produce incompatible results."

B4800 - SN is treated as if the C controller specified UN.

B2900 - SN data type in the C address controller sets
B3900 Invalid Instruction.
B4900

V3 - SN data type in the C address controller causes an Invalid Instruction fault (IEX = 03).

A.13.2 Use of undigits A-F for "M" or values for "Av" not specified may produce incompatible results."

B4800 - Results are undefined and may go undetected.

B2900 - Results in an Invalid Instruction. A partial result
B3900 may have been stored in the "C" field.
B4900

V3 - Causes an Invalid Instruction fault (IEX = 07). A partial result may have been stored in the "C" field.

A.13.3 "Overlap of "A", "B", or "C" fields in any manner may produce incompatible results."

B2900 - Total overlap of "A" and "C" fields will produce
B3900 expected results. Any other form of overlap produces undefined results.

B4800 - If "A" address = "C" address, the "A" and "C" data
B4900 types are the same, and the edit operators in the
V3 "B" field consist of any subset of Move Suppress, Move Digits, and Move Characters, the results are the same as the B2900/3900. All other cases produce undefined results.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.14 TRN

A.14.1 "If AF indicates literal, incompatible results may be produced."

B4800 - Since AF and BF are concatenated, a literal length
B2900   of hundreds of units will result.
B3900

B4900 - Results in an Invalid Instruction.

V3 - Causes an Invalid Instruction fault (IEX = 21).

A.14.2 "If the "C" address controller data type is SN, incompatible results may be produced."

B4800 - SN is treated as if it were specified as UN.

B2900 - Results in an Invalid Instruction.
B3900
B4900

V3 - Causes an Invalid Instruction fault (IEX = 03).

A.14.3 "If the "A" and "C" data types are both UA or both UN, the "A" and "C" fields may totally overlap. All other forms of overlap may produce incompatible results."

B4800 - Results are undefined.

B2900 - Results are undefined and may be different than
B3900   the B4800.

B4900 - Results are unpredictable. (See "Introduction"
V3   note)

A.14.4 B Address restrictions.

B4800 - B Address must be MOD 1000.

B2900 - No address restrictions.
B3900
B4900
V3
V5

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.15 MVR

"Partial overlap of the "A" and "B" fields may produce incompatible results."

B4800 - The first move results in the data between "A" and "B" "smeared". This result is produced BF times in the "B" field.

B2900 - No overlap restrictions. Produces correct result
B3900 in the "B" field.

B4900 - If the "A" address = the "B" address and their
V3 data types are the same, the correct result will be stored. In all other cases, the results are undefined and depend upon the data types, the field lengths, and the module of the addresses.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.16 ARITHMETIC INSTRUCTIONS

"If the operand data contains undigits other than in the sign digit, incompatible results may be produced."

B4800 - Undigits in arithmetic data other than the sign or zone digits are not detected as an error.

B2900 - Undigits in arithmetic data other than the sign or
B3900 zone digis are detected as errors. The entire field containing the invalid unidgits remains unchanged. The processor reports the address of the instruction in error. The Overflow Flag is always set.

B4900 - Undigits in arithmetic data other than the sign or zone digits are detected as errors. If the operand in error is also written (i.e., INC), the operand may be partially overwritten with the new result but the detected undigits will still be present. The processor reports the address of the instruction in error. The Overflow Flag will not be set.

V3 - Undigits in arithmetic data other than the sign or zone digits are detected as errors. If the operand in error is also written (i.e., INC), the operand may be partially overwritten with the new result but the detected undigits will still be present. The processor will report the address of the instruction in error in the resultant Hardware Call procedure. The Overflow Flag will not be set.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.17 RAA, RAS, RSU, RSS, RMU, RMS

If the mantissa of an input operand is not normalized (contains leading zeros), incompatible results may be produced.

B4800 B4900 V3	- The unnormalized data will be used for the operation, which may produce a less precise result than if the data had been normalized.
B2900 B3900	- The data will be normalized prior to the operation.

A.18 RAA, RAS, RSU, RSS

Different processors may maintain differing number of significant digits during the computation, thereby producing slightly incompatible results.

B4800 B4900 V3	- Single precision maintains 9 significant digits; double precision maintains 17.
B2900 B3900	- 215 significant digits are maintained.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.19 NTR

A.19.1 "If AF specifies literal, incompatible results may be produced."

B4800 - Undefined results.

B2900 - Results in an Address Error.
B3900

B4900 - Results in an Invalid Instruction.

V3 - Causes an Invalid Instruction fault (IEX = 21).

A.19.2 "If the address to be written into base relative location 000040 exceeds six digits, incompatible results may be produced."

B4800 - Results in an Address Error.

B2900 - Results in an Invalid Instruction.
B3900
B4900

V3 - Causes an Invalid Instruction fault (IEX = 04).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.20 MVD

A.20.1 "Partial overlap of "A" and "B" may produce incompatible results."

B4800 - If overlap occurs on a move forward and the "A"
B2900 address is less than "B" address, the source field
B3900 digits, in groups of four, are moved to the destination field with smear. If overlap occures on a move backward and the "A" address is greater than "B" address, the source field digits, in group of four, are moved to the destination field.

B4900 - Move Forward
V3

If "A" - "B" < 4, groups of 4 digits will be moved and will replicate accordingly.

Move Backward

The difference between "A" and "B" will be replicated.

A.20.2 "Use of a literal may produce incompatible results."

B4800 - Literals are allowed.

B2900 - Results in an Address Error.
B3900

B4900 - Results in an Invalid Instruction.

V3 - Causes an Invalid Instruction fault (IEX = 21).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.21 MVL

A.21.1 "A literal may produce incompatible results".

B4800	- Literal is allowed.
B2900 B3900	- Literal is allowed but not recommended.
B4900	- Literal will result in an Invalid Instruction.
V3	- Literal will cause an Invalid Instruction fault (IEX = 21).

A.21.2 "Any partial or total overlap may produce incompatible results".

B4800 B4900 V3	- Total overlap allowed on: 1) Identical B and C field or 2) Identical A and B field. No partial overlap is allowed.
B2900 B3900	- There are no overlap restrictions.

A.21.3 "If the three address controllers are not equal, incompatible results may be produced."

B4800	- Not detected. "C" address controller used for all three.
B2900 B3900 B4900	- Results in an Invalid Instruction.
V3	- Causes an Invalid Instruction fault (IEX = 03).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.22 BRT, BST

A.22.1 "Use of literal may produce incompatible results".

B4800 - Literal is allowed.
B2900
B3900

B4900 - Results in an Invalid Instruction.

V3 - Causes an Invalid Instruction fault (IEX = 21).

A.22.2 "If the "A" controller specified SN, incompatable results may be produced."

B4800 - SN treated as if UN was specified.

B2900 - Results in an Invalid Instruction.
B3900
B4900

V3 - Causes an Invalid Instruction fault (IEX = 03).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.23 SRD

"Undigits in AFBF or link address may produce incompatible results."

B4800 B2900 B3900	- There is no undigit check for AFBF or the Link in the R/D area.
B4900	- Undigits are checked for AFBF and the following links will result in an Address Error.
V3	- Undigits in AFBF and the following links will cause an Address Error fault (AEX = 42).

A.23.1 SRD

B4800 B2900 B3900 B4900	- SRD resets Processor Interrupt.
V3	- Processor Interrupt is not reset by this instruction.

A.24 COMPATIBILITY

This instruction is functionally different than the same op code in prior processors.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.25 HBK

"Specification of an AF indirect field length may produce incompatible results."

B4800 - An AF indirect field length may have many levels of indirection. Due to the indirect field length flag bits, the final AFA value must be in the hexadecimal range 0-B. Undigits are allowed in the final AFB value.

B2900 - If the original AF does not specify indirect field
B3900 length, the AF is valid and unchanged. However, if the indirect field length bits are set, the indirect field length is checked for errors and resolved (only one level of indirection is resolved). The final resolved field may be any value and is ignored.

B4900 - Any undigits in the original AF will cause an
V3 Invalid Instruction unless contained in a valid indirect field length or literal specification. If the indirect field length bits are set, the indirect field length is checked for errors and resolved (only one level of indirection is resolved). The final resolved field may be any value and is ignored.

A.25.1 HBK

B4800 - The Halt Digit is located at absolute memory
B2900 address 77.
B3900
B4900

V3 - The Halt Digit is located at absolute memory address 48.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.26 SLL/SLD

A.26.1 "On a successful SLL/SLD the sign digit of B Address (and previous B Address in SLD) may produce incompatible results."

B4800 - The digit stored in sign digit field in IX1 (and IX2 in SLD) is neither a "C" or a "D".

B2900 - Standard EBCDIC sign is stored in IX1 (and IX2 in SLD).
B3900
B4900
V3

A.26.2 "If the "A" address controller specifies SN, incompatible results may be produced."

B4800 - SN treated as if UN was specified.

B2900 - Results in an Invalid Instruction.
B3900
B4900

V3 - Causes an Invalid Instruction fault (IEX = 03).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.29 UNDIGITS IN INTERMEDIATE INDIRECT ADDRESSES

"Undigits in an unresolved intermediate address may produce incompatible results."

B4800	- The undigits will not be detected and will be used as an address which would fetch meaningless data.
B2900 B3900	- Undefined results in final address without resulting in an Address Error.
B4900	- Results in an Address Error.
V3	- Causes an Address Error fault (AEX = 32).

A.30 INDEXING ABOVE LIMIT OR BELOW BASE

"An attempt to index below the BASE or above the LIMIT may produce incompatible results."

B4800 B2900 B3900	- Indexing below the BASE or above the LIMIT such that the final address wraps back around to within the Base/Limit will produce an unpredictable result. Otherwise, it causes an Address Error.
B4900	- Results in an Address Error.
V3	- Causes an Address Error fault (AEX = 11).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.31 MEMORY ERROR REPORT

V3 - The location of the 10 digit field for the Memory Error Report is software controlled. The Memory Error Report Address, previously loaded with a Write Hardware Register instruction (OP = 65:BF = 02), is used as a pointer to the selected memory locations. This address must be mod 10.
The format for this field is as follows:

MSD									LSD
E8	S4	C3	B1	Q2	L4	R16	R12	R08	R04
S7	S3	C2	M3	Q1	L3	R15	R11	R07	R03
S6	S2	C1	M2	G2	L2	R14	R10	R06	R02
S5	S1	B2	M1	G1	L1	R13	R09	R05	R01

E8 = Type of error.
0 = Single bit error.
1 = Multiple bit error.

S1-S7 = Error syndrome (chip location map)

C1-C3 = Memory card location (0 - 7)

B1-B2 = Bank in error
00 = Upper left bank
01 = Upper right bank
10 = Lower left bank
11 = Lower right bank

G1-G2 = Orginating read requester
00 = READER
01 = FORMATTER
10 = XMD
11 = XME

M1-M3 = Memory type
000 = 1.25 MB board
001 = 5 MB board
100 = 20 MB board
101 = 10 MB board

Q1-Q2 = Memory requestor
00 = IOP
01 = WRITE CTL
10 = READ CTL
11 = Reserved

L1-L4 = Logical memory module ID number (0 - 7)

R01-R15 = Reserved

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.32 BASE INDICANT VALUES

"Some values of the Base Indicant digit may be invalid"

V3 - Base Indicant values of "8 - F" are reserved and will cause an Address Error fault (AEX = 13).

A.33 USER SERVICES MEMORY AREA TABLE ENTRY

The first entry in the User Services Memory Area Table describes the environment of the MCP Data Area as follows:

V3 -

INFORMATION	DIGITS
Base Address	00-04
Limit Address	05-09
Software Use	10-19

Note - Lowest memory address = 00

The Base and Limit addresses of the User Services Memory Area Table entry are mod 1,000.

Software must add 10,000 to the desired Base and Limit values to provide absolute memory addresses.

V5 -

INFORMATION	DIGITS
Base Address	00-05
Limit Address	06-11
Software Use	12-13
Memory Area Status Table Number	14-19

The Base and Limit addresses of the User Services Memory Area Table entry are mod 1,000 and do not require any adjustment to provide absolute memory addresses.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.34 MEMORY AREA TABLE ENTRY FORMATS

V3 - The format of each Original entry is as follows:

INFORMATION	DIGITS
Base Address	00-04
Limit Address	05-09
Software Use	10-13
Memory Area Status Table Number	14-19

Note - Lowest memory address = 00

Software must add 10,000 to the desired Base and Limit values to provide absolute memory addresses.

V5 -

INFORMATION	DIGITS
Base Address	00-05
Limit Address	06-11
Software Use	12-13
Memory Area Status Table Number	14-19

Note - Lowest memory address = 00

An Original entry is indicated if the most significant digit of the entry has a value of "0 - 9".

The Base and Limit addresses in an Original entry are mod 1,000 and do not require any adjustment to provide absolute memory addresses.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.34 MEMORY AREA TABLE ENTRY FORMATS (Continued)

V3, V5 - The format of each "C" Copy descriptor entry is as follows:

INFORMATION	DIGITS
Type	00
Reserved	01
Environment Number	02-07
Memory Area Number	08-09
Software Use	10-19

Note - Lowest memory address = 00

The Type digit of a "C" Copy entry is equal to "C".

V3, V5 - The format of each "E" Copy descriptor entry is as follows:

INFORMATION	DIGITS
Type	00
Absolute Address of next chained MAT entry	01-09
Software Use	10-19

Note - Lowest memory address = 00

The Type digit of a "E" Copy entry is equal to "E".

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.34 MEMORY AREA TABLE ENTRY FORMATS (Continued)

V3, V5 - The format of each Memory Area Fault entry is as follows:

INFORMATION	DIGITS
Type	00
Reserved	01
Faulted Area Table Address	02-09
Software Use	10-13
Memory Area Status Table Number	14-19

Note - Lowest memory address = 00

The Type digit of a Memory Area Fault entry is equal to "F".

V3, V5 - The format of each Unused entry is as follows:

INFORMATION	DIGITS
Type	00
Reserved (must be zero)	01-09
Software Use	10-19

Note - Lowest memory address = 00

The Type digit of an Unused entry is equal to "B".

Use of all other Type digit values is reserved and will cause an Invalid Instruction fault (IEX = 50 or 60).

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.35 ABSOLUTE ADDRESSES

V3 - On certain absolute address fields, the size of the "Sub-Base Zero Memory" (currently 10,000) must have been added to each address. These fields are:

1. The eight digit absolute address of the Environment Table in each Reinstate List Entry.

2. The eight digit absolute address of the Memory Area Table in each Environment Table Entry.

3. The Mod 1K Base and Limit fields in each original Memory Area Table Entry.

A.36 TIME OF DAY COUNT RATE

V3 - The Time of Day timer value is incremented by 1000 every millisecond rather than by one every microsecond and the three least significant digits of the time field will be set to zero.

V5 - The Time of Day timer value is incremented by one every microsecond.

A.37 LOCK/UNLOCK

V3 - The Lock Status Field of the Lock Structure will not be used to determine if the Lock is available.

A.38 REINSTATE LIST ENTRY SPECIFICATIONS

V3 - The Reinstate List Entry Size is 200 digits. The Reinstate List may not exceed one million digits in size. The address of the Reinstate List must be mod 1000.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.40 SST

BF = 00 System Status

V3 - The processor will store, maximum, a one byte status indicator. (No padding is required for the remaining 99 bytes.)

BF = 01 System I/D

V3 - The processor will not store the Serial Number or the Firmware Level. The character string "V310", "V340", or "V380" will be stored left justified with blank fill in the Processor Type field depending upon the performance level of the machine. The character string "A" will be stored left justified with blank fill in the Specification Level field.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.43 DIFFERENT REFERENCES ARE RECALCULATED

V3 - The following references will be recalculated:

Current Reinstate List Entry Pointer
MCP Environment Table Address
Number of Entries in the MCP Environment Table.

This variant must be followed by an Interrupt (OP = 90) or a Virtual Branch Reinstate (OP = 93) instruction.

A.44 TASK STATE MAINTAINED WITHIN THE PROCESSOR

V3 - The KERNEL Base/Limit entries are not maintained within the processor.

V5 - The KERNEL Memory Area Table Base/Limit entries are maintained within the processor.

A.45 TASK TIMER FAULT

V3 -
No task timer fault will occur.

V5 -
Task timer fault will occur 99.9999 seconds after the time slice expires.

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.46 REINSTATE LIST

Each Reinstate List entry contains the following information:

V3/V5 - Digit	Purpose
000-007	Link to Next Reinstate List Entry
008	Soft Fault Pending Flag
009	I/O Flags
010-015	Number of Entries in Environment Table
016-018	Environment Table Address Expansion Area
019-027	Environment Table Address
028-037	Failed Hardware Call R/D Area
038-039	Task Processor Priority
040-043	Task Number Owning
044-047	Next Task on List
048-049	State Indicator
050-053	MCP Canonical Lock Number
054-057	User Canonical Lock Number
058-061	Operating Claim
062-070	Next Scheduled Run Time
071-076	Task Wait Time
077-082	New Time Slice
083-090	Direct Time Accumulated
091-092	Mode Indicator Save Area
093-101	Software Usage
102-105	Task Number
106-113	Time Slice Remaining
114-199	Interrupt Frame
114-141	-Accumulator
142-149	-Measurement Register
150-151	-Interrupt Mask
152-183	-Mobile Index Registers
184-185	-Mode Indicators
186-187	-COM & OVF Flags
188-193	-Active Environment Number
194-199	-Instruction Address

Note - The lowest memory address = 00

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.46 REINSTATE LIST (Continued)

The State Indicator field in the Reinstate List Entry contains the following information:

Value	Purpose
00	Runable
01	Waiting Lock
02	Waiting Event
03	Dozing
04	Waiting Termination
05	Failed Hardware Call
06	Kernel Entry
07	Invalid Virtual Branch Reinstate
08-0C	Reserved
0D	Waiting Start-up
0E	Available
0F	Suspended
10-FF	Reserved

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.47 MEMORY AREA STATUS TABLE ENTRY

A Memory Area Status Table entry contains the following information.

V3/V5 -

Digit	Bit	Purpose
00-01		Hardware Lock (For use by processor)
02	3	Memory Area Present
02	2	To be Rolled Out
02	1	Reserved
02	0	I/O Inhibited Memory Area
03-05		Software Usage
06-09		Number of I/O's in Process
10-13		Task Number of Owner
14-19		Environment Number of Original
20-21		Memory Area Number of Original
22-25		Memory Area Size
26-39		Available

Note - The lowest memory address = 00

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.48 ENVIRONMENT TABLE ENTRY

The following description is of an Environment Table entry.

V3 - Digit	Bit	Purpose
00-07		Memory Area Table Address
08-09		Number of Entries in the Memory Area Table
10		Reserved
11		Copy Protection Digit
	3	Reserved
	2	Reserved
	1	Source Copy Enable
	0	Destination Write Enable
12-19		Reserved

Note - The lowest memory address = 00

V5 - Digit	Bit	Purpose
00-08		Memory Area Table Address
09-10		Number of Entries in the Memory Area Table
11		Copy Protection Digit
	3	Reserved
	2	Reserved
	1	Source Copy Enable
	0	Destination Write Enable
12-19		Reserved

Note - The lowest memory address = 00

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.50 TRACE FAULT DATA

The following information is stored on the stack on trace faults:

V3, V5

DIGIT OFFSET WITHIN HARDWARE CALL STACK FRAME	DATA FORMAT	DESCRIPTION
112 - 121	000IAAAAAA	Program counter; I = Base Indicant
122 - 131	0PAFBFXXXX	OPSYL with final AF/BF (fully resolved)
132 - 141	CX0IAAAAAA	Non-literal data operand ASYL
	LLLLLLXXXX	Literal ASYL
	0X0IAAAAAA	Branch Address
142 - 151	CX0IAAAAAA	BSYL - can be other info depending on OP
152 - 161	CX0IAAAAAA	CSYL - can be other info depending on OP
162 - 191		Reserved

C = Address Controller, fully resolved (the least significant two bits specify the operand data type. The remaining bits are unspecified and may contain non-zero values.)

I = Base indicant digit

A = Address digits

L = Unmodified literal

X = Don't care

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.51 MEMORY AREA FAULT HARDWARE CALL PROCEDURE

V3 - Task Number is always zero.

V5 - Task Number is reported accurately.

A.52 IMPROPER UNDIGITS IN BRANCH ADDRESS SYLLABLES

B2900/- Undigits in branch addresses other than the address
B3900 controller or extended digit positions will always cause an address error.

B4900/- Undigits in branch addresses other than the address
V3/V5 controller or extended digit positions will only cause an error if the branch is taken.

A.53 MEMORY ADDRESSABILITY

B4800 - This machine can address 10 million digits. Only 2 million could be physically attached.

B2900/- This machine can address and have physically
B3900/ attached 10 million digits.
B4900

V3 - This machine can address 100 million digits. Only 80 million could be physically attached.

V5 - This machine can address 1 billion digits. Up to 640 million digits can be physically attached per processor cabinet.

INFORMATION	DIGITS
Constant-1	00-05
"A" Absolute Address	06-13
"B" Absolute Address	14-21
Constant-2	22-29

Note - The lowest memory address = 00

APPENDIX A - COMPATIBILITY NOTES (Continued)

A.54 HANDLING OF THE SIGN DIGIT IN ILD/IST/RLD/RST

B4800 - Undigits may be loaded and recovered from the sign digit of the accumulator.

B2900/- B3900/ B4900/ V3/ V5 All digits loaded into the sign digit of the accumulator will be stored as a "C" or "D" in the accumulator and will be recovered as such via a Store instruction.

INFORMATION	DIGITS
Constant-1 (Moved)	00-05
"A" Address (Converted)	06-13
"B" Address (Converted)	14-21
Constant-2 (Moved)	22-29
Memory Area Status Table Number	30-35

Note - The lowest memory address = 00