NCR CONTROL DATA

ADVANCED SYSTEMS LABORATORY

Document Title: GDS, IPL Processor-Memory, Model Independent

Originator/Author: W. H. Specker

Doc. Date: 2/28/75

Doc. No.(if applicable) ASL00211 Revision (if appl.) D.

Design Data Base Status: A.3.1

Applicable Project/Product:

APPROVAL SIGNATURES:		DATE:
Originator:	W. H. Specker	2/28/75
Project Manager:	G. M. Schumacher	2/28/75
General Manager,ASL Prog.Ofce:	J. A. Kershaw	2/28/75
General Manager,ASL-C:	R. O. Gunderson	2/28/75
General Manager,ASL-W:	T. F. Belden	
Director and General Manager,ASL:	D. L. Slais	
Other:		

Disclaimer: This document is a working paper only and does not necessarily represent any official intent on the part of National Cash Register Company, nor of Control Data Corporation, nor of the Advanced Systems Laboratory.

THIS DOCUMENT CONTAINS INFORMATION PROPRIETARY TO THE NATIONAL CASH REGISTER COMPANY AND CONTROL DATA CORPORATION. ITS CONTENTS SHALL NOT BE DIVULGED OUTSIDE OF EITHER COMPANY NOR REPRODUCED WITHOUT EXPLICIT PERMISSION OF THE DIRECTOR AND GENERAL MANAGER, NCR/CDC ADVANCED SYSTEMS LABORATORY.

NCR / CDC PRIVATE

CONTROL DATA

DELIVER TO . . .

FIRST NAME [redacted] INITAL LAST NAME

DEPT. [redacted] TTOFAC

MAIL STATION NO. | ROOM NO. | EXT.

FROM

FIRST NAME INITAL LAST NAME

Janet Knight

DEPT | FACILITY CODE

Advanced Sys. Lab. | PGAASL

MAIL STATION NO. | ROOM NO. | EXT.

6420

COMMENTS:

NOTE TO SENDER: IF INFORMATION IS OMITTED DELIVERY COULD BE DELAYED OR MAIL RETURNED.
* CAN BE FOUND IN CORPORATE TELEPHONE DIRECTORY.

☐ AIR MAIL REQUIRED

AA361 REV. 1/71

COPY NUMBER 121

A.3.1

DOCUMENT	SECTION

NCR CONTROL DATA

ADVANCED SYSTEMS LABORATORY

	THIS	REPLACES
FILE		
DATE	Feb. 28, 1975	Dec. 9, 1974
PAGE	1	1

APPROVED ASL	APPROVED NCR	APPROVED CDC

IPL

General Design Specification

Model Independent

Processor/Memory

Doc. No. ASL00211
Revision D

NCR / CDC PRIVATE

Table of Contents Page No.

NCR/CDC PRIVATE

Table of Contents (continued) Page No.

NCR/CDC PRIVATE

Table of Contents {continued}

NCR/CDC PRIVATE

Table of Contents {continued}

NCR/CDC PRIVATE

NCR/CDC PRIVATE

NCR/CDC PRIVATE

NCR / CDC PRIVATE

NCR / CDC PRIVATE

NCR / CDC PRIVATE

NCR / CDC PRIVATE

NCR / CDC PRIVATE

1.0 INTRODUCTION

1.1 Scope

This General Design Specification is intended to define the common properties and characteristics of Processor Models P1 through P4, and Central Memory Models, M2 through M4 which constitute major firmware/hardware components of the Integrated Product Line. Included in this model independent specification is the description of the Virtual Memory Mechanism commonly applicable to these major system components.

1.2 Applicable Documents

IPL Architectural Definition
IOSS Model Independent GDS
IPLOS GDS

1.3 Configurations

The IPL architecture shall allow substantial flexibility in the interconnection of the basic elements of an IPL computer system. These elements shall consist of central processors, central memories and I/O subsystems.

For purposes of this specification, IPL processors will be referred to as processors (P). Where differentiation between the four models is required, they will be referred to as P1, P2, P3, or P4.

Likewise, central memory will be referred to as memory (M). Where differentiation between the three models is required, they will be referred to as M2, M3 or M4.

The Processors and their associated memories will be divided into two categories: lower IPL and upper IPL.

The lower IPL processor, designated as P1, is intended to be used primarily as an I/O processor in configurations which include one or more of the Processors, P2, P3, or P4.

1.3.1 Interelement Transfer Paths

All data transfers between processors shall be via central memory.

NCR/CDC PRIVATE

Transmission of data between central memories (M2, M3 or M4) and upper IPL central processors (P2, P3 or P4) shall occur over compatible, 64-bit wide data buses. Enforcement of bus compatibility across the three models will allow any of these processors (P2, P3, P4) to be connected to any of the memories (M2, M3, M4).

Transmission of data between central memories (M2, M3, M4) and the lower IPL processor, (P1), shall likewise occur over compatible data buses with a reduction in width to 32 bits. Compatibility in the design of these buses will allow the lower IPL processor, (P1), to be connected to any central memory (M2, M3 or M4).

1.3.2 Interelement Transfer Rates

The maximum transfer rate over the data buses shall be 64 bits or 32 bits every 56 nsec (read or write) for a 64-bit or a 32-bit data bus, respectively. This maximum data transfer rate shall be determined by central memory and shall be the same for all three central memory models; M2, M3 and M4. The difference between these models shall be primarily in capacity, number of available ports, and the number of banks for minimizing conflicts.

The characteristics of each central memory model are summarized as follows:

Model	Capacity M bytes	64-bit Ports	32-bit Ports	Number of Banks
M2	1-4	2	4	8-16
M3	2-8	4	8	16-32
M4	4-16	4	16	32-64

1.3.3 Interelement Connection Alternatives

Each of the upper IPL central processors (P2, P3 or P4) shall provide the means for accommodating two 64-bit ports for attaching two memory buses. This feature will allow for a processor to have either a redundant (backup) data path to the same memory or individual connections to two independent memories.

For the same reason, the lower IPL processor, P1, shall provide the means for accommodating two 32-bit ports for attaching two memory buses.

NCR/CDC PRIVATE

Each of the IPL processors (P1 through P4) shall provide the means for accommodating two interfaces for connection to two Service Processors (SPs); one active, the second as backup. The SP shall serve as the programmable maintenance facility for these processors.

The multiple 64-bit and 32-bit ports within each of the memory models provides the means for several processors to share the same memory, thus allowing a substantial degree of flexibility in memory and processor configuration alternatives. Typical examples are illustrated in Figures 1.3-1, 1.3-2 and 1.3-3.

Note: The maximum one-way, electrical distance between interconnected IPL processors and central memories shall not exceed 2 clock cycles of propagation delay for 64-bit buses and shall not exceed 3 clock cycles of propagation delay for 32-bit buses. See 4.7.

1.4 General Timing Considerations

Within each processor, instruction execution shall be "conceptually serialized." Although central memory and register references may occur out of order, (to whatever degree required by a processor's model — dependent implemention in the achievement of its cost/performance goals), the results from each of the associated instructions, as observed by the processor performing their execution, shall be the same as if such instructions were actually executed in a serialized fashion (i.e., each instruction's execution would be completed before the execution of any subsequent instructions would begin.) The single exception to this concept shall occur in the case of self-modifying programs as stated in paragraph 2.1.2 of this specification.

Processor operations shall be further serialized, as observed by other processors, only to the extent that the function referred to as "serialization" is included within the execution of certain instructions as described in section 2.6 of this specification.

Program interruptions shall occur between the execution of instructions, and with timing precision relative to the cause of such interruptions, to the extent specified in section 2.8 of this specification.

NCR/CDC PRIVATE

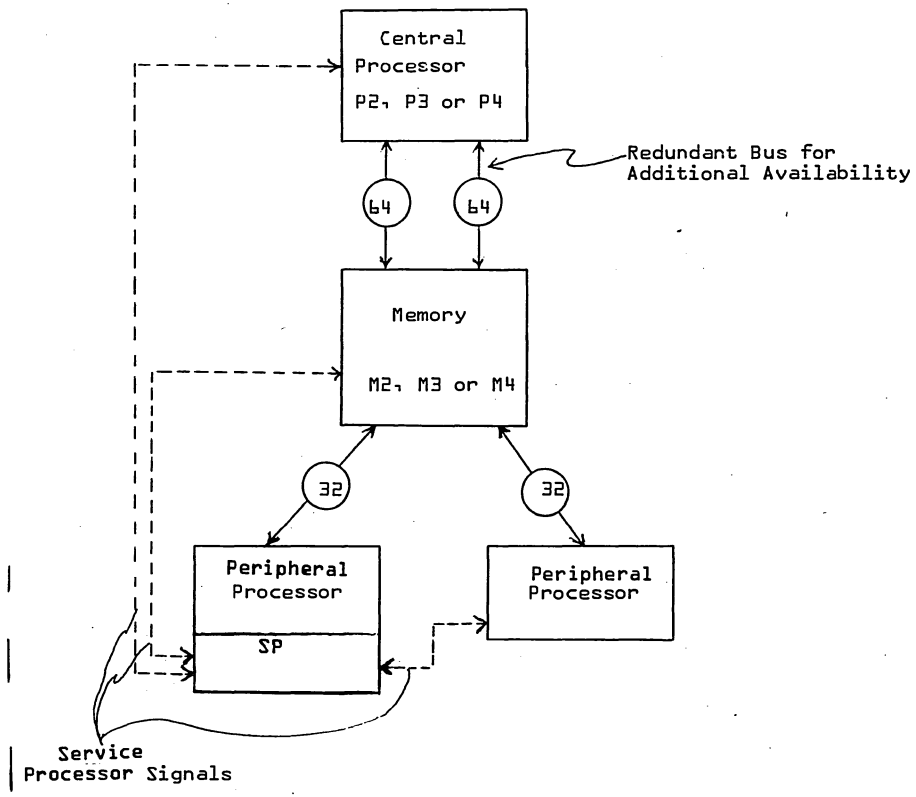


Figure 1.3-1: Example of a Single Central Processor Configuration

Note: See 2.10.1.1 for a description of processor port selection with respect to redundant paths to central memory.

NCR/CDC PRIVATE

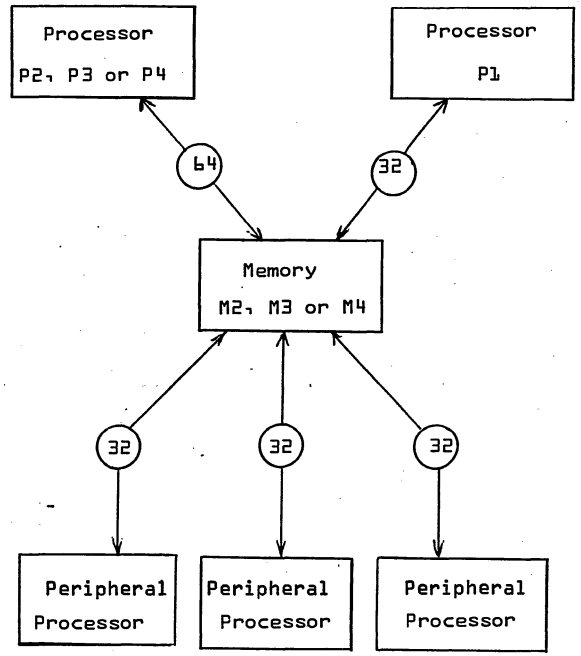


Figure 1.3-2: Example of a Dual Central Processor Configuration

NOTE: SP interfaces are not shown

NCR/CDC PRIVATE

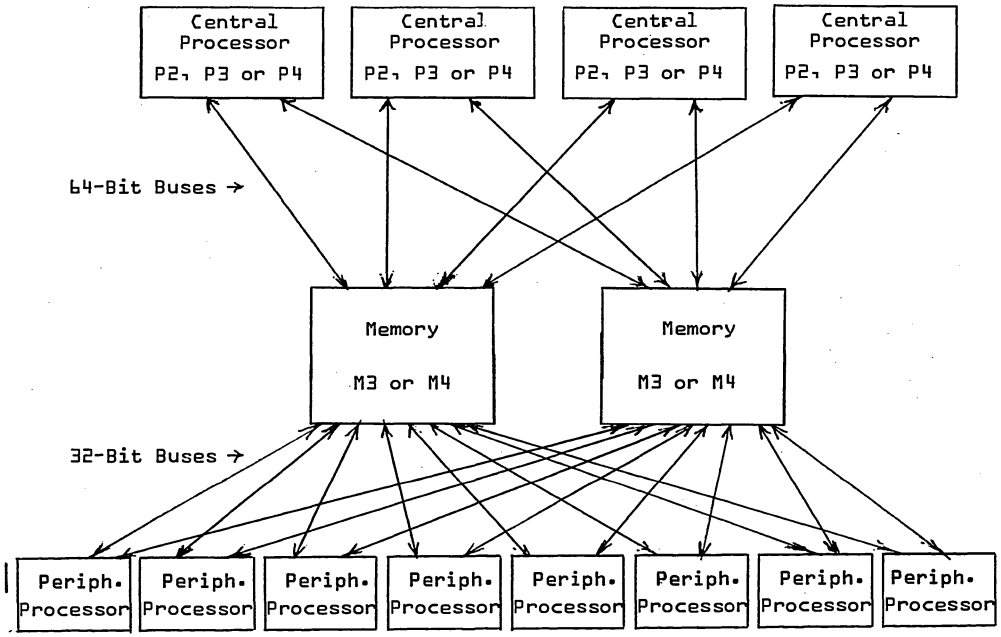


Figure 1.3-3: Example of a Multiple Central Processor System

Note: SP interfaces are not shown.
See 2.10.1.1 for a description of central memory unit selection.

NCR/CDC PRIVATE

2.0 Processor

Processor Models P1 through P4 shall provide the means for reading and translating each of the instruction codes contained in the instruction repertoire, as well as performing the corresponding execution of these instructions as defined by the descriptions contained in this specification.

In order to accomplish instruction fetch and execution, each processor shall additionally provide the means for referencing central memory. Central memory references shall be performed either in virtual mode, which shall include the address translation and protection facilities as described in Sections 3.0 through 3.6 of this specification, or in real mode which shall involve neither address translation nor protection facilities as described in subparagraph 2.1.1.1 of this specification.

2.1 General Description

For the purposes of this specification the operation codes from the instruction repertoire shall be divided into four groups of instructions referred to as the General Instructions, the Business Data Processing Instructions, the Floating Point Instructions, and the System Instructions. In addition to central memory, addressed in virtual or real mode, the execution of the instructions within the first three of these instruction groups, namely the General, BDP, and Flt. Pt. Instructions, shall require the means to reference general containers referred to as the P Register, the A Registers, and the X Registers. Also, the means for detecting and indicating exceptional conditions, which may occur in the course of executing these instructions, shall be provided in accordance with the appropriate instruction descriptions contained in this specification.

The fourth group, namely the System Instructions, shall additionally require the means to reference special containers referred to as the Processor State Registers, Process State Registers, and Memory Maintenance Registers in accordance with the appropriate descriptions contained within sections 2.5 and 2.6 of this specification.

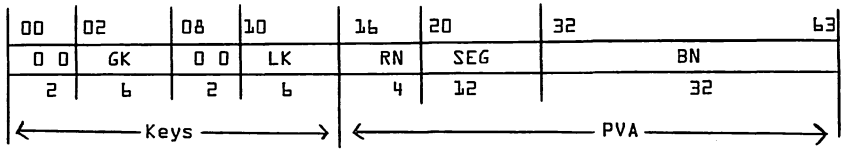
2.1.1 General Registers

The means for referencing a total of 33 General Registers shall be provided.

NCR/CDC PRIVATE

2.1.1.1 P Register

The Program Address Register, referred to simply as the P Register, shall consist of 64 bits, numbered from left to right, beginning with bit position 00. Conceptually, the P Register shall contain the Process Virtual Address, PVA, of an instruction in central memory during the time it is read, interpreted, and executed by the processor. Similarly, the P Register shall contain "Keys" to central memory during each instruction's execution. The contents of the P Register shall be formatted as follows: where the RN (Ring Number), SEG (Segment) and BN (Byte Number) fields are individually described within Section 3.2 of this specification, and the GK (Global Key) and LK (Local Key) fields are individually described within paragraph 3.6.3 of this specification, with respect to virtual addressing mode.



With respect to real addressing mode, only the BN field from the P Register shall be used. As a result of not utilizing the GK, LK, RN and SEG fields, neither address translation nor protection checking shall be performed and the BN field from the PVA shall be used directly as the real memory address (RMA). The format of an RMA is described in Paragraph 3.1.3 of this specification.

2.1.1.2 A Registers

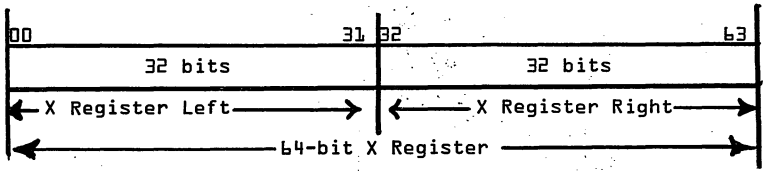
The sixteen A Registers, referred to as the A0 Register through the AF Register (using hexadecimal notation), shall consist of 48 bits each, identical in format to the rightmost 48 bits of the P Register as just previously described, with respect to both real and virtual addressing modes.

Note. Although these address registers are intended for general use in explicitly supplying such PVA's as may be required for branch (jump) and operand references to central memory, an aggregate of eight A Registers, (namely, A0 through A7), shall be implicitly utilized during BDP and CALL instruction executions as described in Sections 2.3 and 2.6, respectively, of this specification.

NCR/CDC PRIVATE

2.1.1.3 X Registers

The sixteen X Registers, referred to as the X0 Register through the XF Register (using hexadecimal notation), shall consist of 64 bits each with their bit positions numbered from left to right, beginning with bit position 00, as follows:



The 64-bit contents of an X Register may be treated as a logical quantity, a signed binary integer, or a signed floating point number. Bit string, byte string, 32-bit halfword (right-justified in bit positions 32 through 63), and 64-bit word operations shall be provided for the contents of the X Registers.

Note. Although these operand registers are intended for general use in explicitly supplying such operands as may be required for accomplishing the execution of a majority of instructions, the first two X Registers, (namely, X0 and X1), shall be implicitly utilized during certain instructions which require additional input arguments or execution results. In these cases, Register X0 Right shall normally be used to supply additional input parameters to instruction execution and Register X1 Right shall be utilized to receive additional results from instruction execution. Whenever applicable, the instruction descriptions contained in this specification will fully define all register utilizations which shall be implicit in nature, including those cases in which the contents of Register X0 shall be interpreted as consisting, partially or entirely, of zeros.

2.1.2 Instructions

Instructions shall be 16 bits or 32-bits in length, according to one of the six formats described in the following sub-paragra

Programmed modification of the instructions comprising a stored program in central memory may lead to undefined results.

2.1.2.1 Formats j k i D and S j k i D

Operation Code	j	k	i	D
8	4	4	4	12

Operation Code	S	j	k	i	D
5	3	4	4	4	12

For these 32-bit instruction formats: the j, k, and i fields shall provide register designations, the D field shall provide either a signed shift count, a positive displacement or a bit-string descriptor, and the S field shall provide a sub-operation code.

2.1.2.2 Format j k

Operation Code	j	k
8	4	4

For this 16-bit instruction format, the j field shall provide a register designation, a sub-operation code, or an immediate operand value and the k field shall provide a register designation.

2.1.2.3 Formats j k Q and S j k Q

Operation Code	j	k	Q
8	4	4	16

Operation Code	S	j	k	Q
5	3	4	4	16

For these 32-bit instruction formats, the j and k fields shall provide register designations or sub-operation codes. The 16-bit Q field shall provide a signed displacement or an immediate operand value. The S field shall provide a sub-operation code.

2.1.2.4 Format j k Q/2

Operation Code	j	k	Q
8	4	4	16

For this 32-bit instruction format, the j and k fields shall provide 2 register designations each and the 16-bit Q field shall provide 2 separate displacement values of 8 bits each. This instruction format shall be used exclusively by the instructions in the BDP Instruction group and the definition of the manner in which the fields from instructions of this format are used, is detailed in Section 2.3 of this specification.

2.1.2.5 Access

Instruction accesses shall be confined to byte addresses which are 0, modulo 2. Thus, values which have a one bit in position 63 shall be detected at the time an attempt is made to transfer such values into the P Register, and Address Specification error shall be detected, and the corresponding program interruption shall occur.

For the purpose of establishing central memory access validation, the reading of every instruction shall be an Execute type access. When specifically included within an instruction's description, the appropriate central memory access performed for the purpose of fetching the instruction to be subsequently executed, shall be execute validated, provided such a reference occurs in virtual addressing mode. Execute type accesses shall use the ring number contained in the P Register for access validation.
The access validation procedure, which requires the classification of central memory accesses into read, write, execute, and call types, is described in Section 3.6 of this specification. As part of the Virtual Memory Mechanism, this validation

procedure is intended to provide hardware assistance in satisfying the requirements for privacy and protection of information stored in central memory, while simultaneously sustaining the ability of various processes to share central memory information to varying degrees, for varying purposes.
With respect to "demand page" interrupts (Page Table Search without Find conditions as described in subparagraph 2.8.1.10 of this specification) the fetching of an instruction shall be considered as part of that instruction's execution. This shall apply even when the instruction fetch is immediately preceded by a branch exit (as described in paragraph 2.2.3 of this specification) on the part of the previous instruction. Thus, with respect to demand paging, the execution of an instruction shall never include the fetching of the next instruction to be executed.

2.1.2.6 Unused Bits
When one or more bits from an instruction are unused, i.e., their value(s) and associated function(s) are not specified within the instruction description, the execution of these instructions shall not be affected by the values of these bits. However, it is recommended that such bits are equal to zeroes.

2.1.2.7 Nomenclature

Throughout the instruction descriptions contained in this specification, the following conventions shall be used with respect to nomenclature.

a. The expressions "Register Aj" and "the Aj Register" shall be used interchangeably to denote the 48-bit A Register specified by the 4-bit j field from an instruction. Thus, "Aj" shall denote one of the sixteen A Registers, A0 through AF (in hexadecimal notation) corresponding to j field values of 0 through 15 (in decimal notation), respectively.

   The 4-bit k field from an instruction shall be interpreted in a manner identical to the j field (as just described) with respect to the interchangeable expressions "Register Ak" and "the Ak Register."

b. The expressions "Register Xj" and "the Xj Register" shall be used interchangeably to denote the 64-bit X Register specified by the 4-bit j field from an instruction. Thus, "Xj" shall denote one of the sixteen X Registers, X0 through

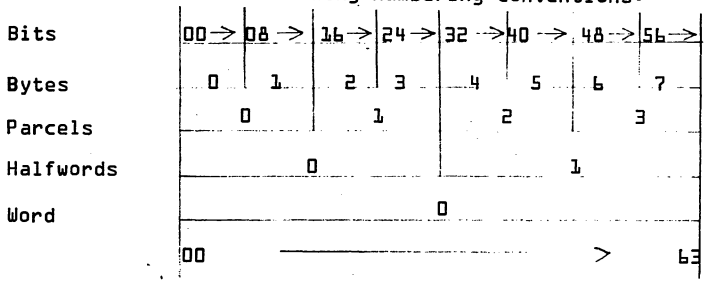
XF (in hexadecimal notation) corresponding to j field values of 0 through 15 (in decimal notation), respectively.

The 4-bit k field from an instruction shall be interpreted in a manner identical to the j field (as just described) with respect to the interchangeable expressions "Register Xk" and "the Xk Register."

c. With respect to the X Registers, the terms "Left" and "Right" shall be used to denote the leftmost and rightmost 32-bit positions, respectively. Thus, "Register Xk Left" shall denote the leftmost 32-bit positions, 00 through 31, of the Xk Register and "Register Xk Right" shall denote the rightmost 32-bit positions, 32 through 63, of the Xk Register.

d. Parentheses shall be used within instruction names to denote "the contents of".

e. Units of information shall be referred to as bytes (8 bits), parcels (16 bits), halfwords (32 bits) or words (64 bits) with the following numbering conventions:



Note: Alphanumeric (including decimal) and floating point data formats are illustrated in Sections 2.3 and 2.4, respectively of this specification.

2.1.3 Address Arithmetic

Address arithmetic operations, referred to as "indexing" and "displacement," shall be performed on signed, 32-bit integers using 2's complement addition without overflow detection.

NCR/CDC PRIVATE

2.1.4 Address Exception

When the leftmost bit of the BN field, (position 32), in any PVA is equal to a one at the time it is used to access central memory, an Address Specification error shall be detected, the central memory access shall be inhibited, and the corresponding program interruption shall occur.

2.1.5 Instruction Reference Numbers

Note: The descriptions for all instructions, or sub-groups of instructions, shall include the name, reference number and field designators for each individual instruction.

Reference numbers for each instruction shall consist of three digits and shall correspond to the associated instruction's entry in Appendix A of this specification.

2.2 General Instructions

For the purpose of this specification, the instructions comprising the General Instruction group shall be further classified, according to function, as described by the titles for paragraph numbers 2.2.1 through 2.2.11 of this specification.

For the applicable instructions in this subgroup, as well as the instructions in the BDP group, (Section 2.3), "false" exception conditions shall not be detected as a result of interpreting any PVA for which the associated central memory data field has a length equal to zero.

2.2.1 Load and Store

This sub-group of instructions shall provide the means for transferring data, in the form of a single bit, a byte string, a 64-bit word, or multiple 64-bit words between one or more Registers and one or more locations in central memory as specified by the individual operation codes.

For the purpose of establishing operand access validity for the associated central memory read and write accesses, the ring number used for validation shall be the value of the ring number contained in bit positions 16 through 19 of the associated A Register. Access validation shall occur in Virtual Addressing Mode only.

The central memory operand access types shall be read-access for any instruction which loads an A or X register, and write-access for any instruction which stores an A or X register.

NCR/CDC PRIVATE

Instructions which transfer data from one or more Registers to central memory, {namely, Store instructions}, shall not alter the contents of any Register which serves as a source of the data to be transferred to central memory.

2.2.1.1 Load/Store Bytes, XK; Length Per S

a. Load Bytes to Xk from {Aj} displaced by D and indexed by {Xi} Right, Length Per S
001 SjkiD

b. Load Bytes to Xk from {Aj} displaced by Q, Length Per S
002 S jkQ

c. Store Bytes from Xk at {Aj} displaced by D and indexed by {Xi} Right, Length Per S
003 S jkiD

d. Store bytes from Xk at {Aj} displaced by Q, Length Per S
004 S jkQ

Operation: These instructions shall transfer a field of bytes between Register Xk and a byte field in central memory. The direction of transfer is determined by the operation code, and the length of the byte field to be transferred shall be determined by adding one to the value obtained from the S-field of the instruction. The bytes in Register Xk shall be right-justified, so that the appropriate left-most byte positions in Register Xk shall be cleared for load instructions with lengths less than eight, and the appropriate left-most byte positions within the Xk Register shall not be transferred for store instructions with lengths less than eight.

No QI Novabyte

Addressing: The beginning {the leftmost byte position} of the byte string in central memory shall be determined by means of the PVA obtained from the Aj Register, modified by a byte item count determined as follows:

Displacement and Indexing: The 32-bit halfword obtained from register Xi Right and the 32-bit quantity obtained by left-extending the D field with zeroes shall be added to the rightmost 32 bits of the PVA obtained from the Aj Register. In this context, the contents of the X0 Register shall be interpreted as consisting of all zeroes.

Displacement: The Q field from the instruction shall be expanded to 32 bits by means of sign extension and the 32-bit result shall be added to the rightmost 32 bits of the PVA obtained from the Aj Register.

2.2.1.2 Load/Store Word, Xk

a. Load Xk from {Aj} displaced by 8*D and indexed by 8*{Xi} Right
005 jkiD

b. Load Xk from {Aj} displaced by 8*Q
006 jkQ

c. Store Xk at {Aj} displaced by 8*D and indexed by 8*{Xi} Right
007 jkiD

d. Store Xk at {Aj} displaced by 8*Q
008 jkQ

Operation: These instructions shall transfer a word between Register Xk and a word location in central memory. The direction of transfer shall be determined by the operation code.

Addressing: The item location in central memory shall be determined by means of the PVA obtained from register Aj modified by a 32-bit quantity calculated as follows:

Displacement and Indexing: The 32-bit halfword obtained from register Xi Right shall be shifted left 3 bit positions, end-off with zeroes inserted; the 12-bit quantity obtained from the D field of the instruction shall be expanded to 29 bits by inserting zeroes on the left and shall then be shifted left 3 bit positions with zeroes inserted on the right. The two 32-bit quantities resulting from these operations shall then

be added to the rightmost 32 bits of the PVA obtained from the Aj register. In this context, the contents of register X0 shall be interpreted as consisting of all zeroes.

Displacement: The q field from the instruction shall be expanded to 29 bits by means of sign extension, and shall then be shifted left 3 bit positions with zeroes inserted on the right. The 32-bit result shall then be added to the rightmost 32 bits of the PVA obtained from the Aj register.

Notes: Unless the PVA from the Aj Register consists of a byte address which is 0 modulo 8, an Address Specification error shall occur, the loading or storing of the Xk register shall be inhibited, and the corresponding program interruption shall take place.

2.2.1.3 Load/Store Bytes, Xk, Length Per X0

a. Load Bytes to Xk from (Aj) displaced by D and indexed by (Xi) Right, Length Per X0
   009 jkiD

b. Load Bytes to Xk from (Aj) displaced by q, Length Per X0
   010 jkq

c. Store Bytes from Xk at (Aj) displaced by D and indexed by (Xi) Right, Length Per X0
   011 jkiD

d. Store Bytes from Xk at (Aj) displaced by q, Length Per X0
   012 jkq

Operation: These instructions shall transfer a field of bytes between Register Xk and a byte field in central memory with the direction of the transfer determined by the operation code. The length of the byte field shall be determined by the contents of Register X0 Right.

When the length is equal to zero, these instructions shall result in no operation. When the length is greater than eight, an Instruction Specification error shall be detected, loading or storing of

NCR/CDC PRIVATE

the Xk Register shall be inhibited, and the corresponding program interruption shall occur. The bytes in Register Xk shall be right-justified so that the appropriate leftmost byte positions within the Xk Register shall be cleared for load instructions with lengths less than eight but greater than zero, and the appropriate leftmost byte positions within the Xk Register shall not be stored for store instructions with lengths less than eight.

Addressing: Identical to that described in Section 2.2.1.1

2.2.1.4 Load Bytes, Xk, Length Per j

a. Load Bytes to Xk from (P) displaced by q, Length per j
   013 jkq

Operation: This instruction shall transfer a field of bytes from central memory to register Xk. The length of the byte field shall be determined by the j-field. In all other respects, the operation is identical to that described in 2.2.1.3.

Addressing: The beginning (the leftmost byte position) of the byte field in central memory shall be determined by expanding the q field to 32 bits by means of sign extension and then adding the result to the rightmost 32 bits of the PVA obtained from the P Register.

2.2.1.5 Load/Store Bit, Xk

a. Load Bit to Xk from (Aj) displaced by q and bit indexed by (X0) Right
   014 jkq

b. Store Bit from Xk at (Aj) displaced by q and bit indexed by (X0) Right
   015 jkq

NCR/CDC PRIVATE

Operation: These instructions shall transfer a single bit between Register Xk Right, bit position 63, and a bit position in central memory, with the direction of the transfer determined by the operation code. Additionally, the load instruction shall clear the Xk Register in its leftmost 63 bit positions, 00 through 62.

Addressing: The byte in central memory, containing the bit position to be loaded from or stored into, shall be addressed by means of the PVA contained in the Aj Register modified as follows: The 32-bit halfword obtained from Register X0 Right shall be shifted right three bit positions, end-off with sign extension on the left, and the q field from the instruction shall be expanded to 32 bits by means of sign extension. These two 32-bit results shall then be added to the rightmost 32 bits of the PVA obtained from the Aj Register.

Bit Selection: The bit position within the addressed byte in central memory shall be selected by means of the rightmost three bits obtained from Register X0 Right, bit positions 61 through 63. Values from 0 through 7 for these three bits shall select the corresponding bit position, 0 through 7 within the central memory byte.

Note: The instruction which transfers a bit to central memory shall accomplish the associated central memory operations in a non-preemptive manner, i.e., the byte containing the bit to be stored shall be read, modified in the appropriate bit position to the extent required, and then written such that no other accesses to the addressed byte shall be permitted between these read and write accesses. Moreover, those processors having a Cache, (See 2.9), shall bypass it with respect to the read access and shall purge the associated entry from it with respect to the write access.

When the instruction which transfers a bit to central memory is executed in virtual addressing mode, operand access validation shall consist of write access validation only.

2.2.1.6 Load/Store Ak

a. Load Ak from (Aj) displaced by D and indexed by (Xi) Right
016 jkiD

b. Load Ak from (Aj) displaced by q
017 jkq

c. Store Ak at (Aj) displaced by D and indexed by (Xi) Right
018 jkiD

d. Store Ak at (Aj) displaced by q
019 jkq

Operation: These instructions shall transfer six bytes between the Ak register, right-justified, and a six byte field in central memory, with the direction of transfer determined by the operation code.

Addressing: The left-most byte position of the six byte field in central memory shall be addressed by means of the PVA initially contained in register Aj, modified by a byte item count, in a manner identical to that described in section 2.2.1.1.

Special Load Conditions: The instructions which load Register Ak shall unconditionally transfer only the rightmost 44 bits of the six byte field from central memory to bit positions 20 through 63 of Register Ak.

When this instruction is executed in virtual addressing mode, the larger value of, 1) the leftmost 4 bits of the six byte field from central memory, 2) the leftmost 4 bits in bit positions 16 through 19 of the Aj Register and 3) the R1 field contained in the 4-bit positions 08 through 11 of the segment descriptor associated with the PVA obtained from Register Aj, shall be transferred to bit positions 16 through 19 of Register Ak.

(For the format of a segment descriptor and the definition of its R1 field, see paragraphs 3.3.1 and 3.6.2 of this specification, respectively.

Operation: These instructions shall transfer a single bit between Register Xk Right, bit position 63, and a bit position in central memory, with the direction of the transfer determined by the operation code. Additionally, the load instruction shall clear the Xk Register in its leftmost 63 bit positions, 00 through 62.

Addressing: The byte in central memory, containing the bit position to be loaded from or stored into, shall be addressed by means of the PVA contained in the Aj Register modified as follows: The 32-bit halfword obtained from Register X0 Right shall be shifted right three bit positions, end-off with sign extension on the left, and the Q field from the instruction shall be expanded to 32 bits by means of sign extension. These two 32-bit results shall then be added to the rightmost 32 bits of the PVA obtained from the Aj Register.

Bit Selection: The bit position within the addressed byte in central memory shall be selected by means of the rightmost three bits obtained from Register X0 Right, bit positions 61 through 63. Values from 0 through 7 for these three bits shall select the corresponding bit position, 0 through 7 within the central memory byte.

Note: The instruction which transfers a bit to central memory shall accomplish the associated central memory operations in a non-preemptive manner, i.e., the byte containing the bit to be stored shall be read, modified in the appropriate bit position to the extent required, and then written such that no other accesses to the addressed byte shall be permitted between these read and write accesses. Moreover, those processors having a Cache, (See 2.9), shall bypass it with respect to the read access and shall purge the associated entry from it with respect to the write access.

NCR/CDC PRIVATE

2.2.1.6 Load/Store Ak

a. Load Ak from (Aj) displaced by D and indexed by (Xi) Right
   016 jkiD

b. Load Ak from (Aj) displaced by Q
   017 jkQ

c. Store Ak at (Aj) displaced by D and indexed by (Xi) Right
   018 jkiD

d. Store Ak at (Aj) displaced by Q
   019 jkQ

Operation: These instructions shall transfer six bytes between the Ak register, right-justified, and a six byte field in central memory, with the direction of transfer determined by the operation code.

Addressing: The left-most byte position of the six byte field in central memory shall be addressed by means of the PVA initially contained in register Aj, modified by a byte item count, in a manner identical to that described in section 2.2.1.1.

Special Load Conditions: The instructions which load Register Ak shall unconditionally transfer only the rightmost 44 bits of the six byte field from central memory to bit positions 20 through 63 of Register Ak.

When this instruction is executed in virtual addressing mode, the larger value of, 1) the leftmost 4 bits of the six byte field from central memory, 2) the leftmost 4 bits in bit positions 16 through 19 of the Aj Register and 3) the R1 field contained in the 4-bit positions 08 through 11 of the segment descriptor associated with the PVA obtained from Register Aj, shall be transferred to bit positions 16 through 19 of Register Ak.

(For the format of a segment descriptor and the definition of its R1 field, see paragraphs 3.3.1 and 3.6.2 of this specification, respectively.

NCR/CDC PRIVATE

When this instruction is executed in real addressing mode, the larger value of the leftmost 4 bits of the six byte field from central memory and the leftmost 4 bits of the Aj Register, shall be transferred to bit positions 16 through 19 of the Ak Register.

When the leftmost 4 bits of the six byte field from central memory are all equal to zero, a Ring Number Zero condition shall be detected and, following the completion of the associated Load instruction's execution, the corresponding program interruption shall take place.

NCR/CDC PRIVATE

2.2.1.7 Load/Store Multiple Registers

a. Load Multiple Registers from (Aj) displaced by B*k, Selectivity per (X0) Right
   020 jk

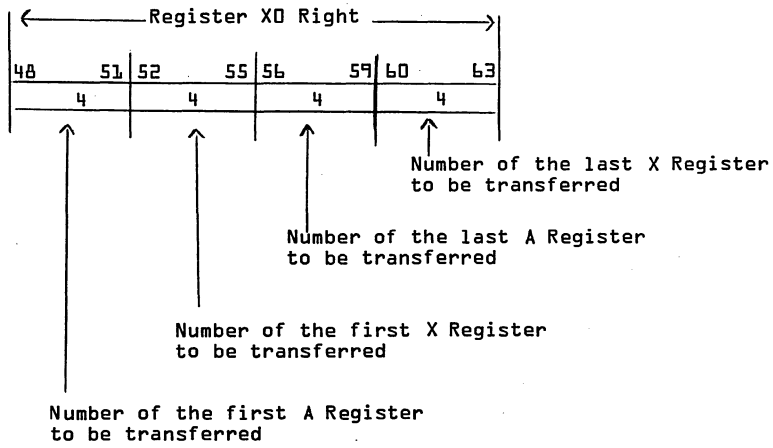
b. Store Multiple Registers at (Aj) displaced by B*k, Selectivity per (X0) Right
   021 jk

Operation. These instructions shall transfer data between the general registers and central memory with the direction of the transfer determined by the operation code. Central memory address formation and general register selections shall be performed as follows:

Address Formation. The beginning address in central memory, of the contiguous word locations to which or from which, as determined by the operation code, the designated transfers shall take place, shall be formed by means of displacement addressing. The 4-bit k field from the instruction shall be expanded to 29 bits by extending leftmost zeroes, these 29 bits shall be shifted left three bit positions with zeros inserted on the right, and this 32-bit shifted result shall be added to the rightmost 32 bits of the PVA initially contained in the Aj Register. The resulting PVA shall be used as the beginning address of the word field in central memory referenced by these instructions.

Register Selection. Selectivity of transfers between general registers and central memory shall be accomplished by interpreting the rightmost 16-bits initially contained in Register X0 Right as four fields of 4-bits each in the following manner:

NCR/CDC PRIVATE



When the first register number is greater than the associated last register number, none of the registers from the corresponding A or X Register groups shall be loaded or stored.

Transfers between registers and central memory shall begin with the A Register Group and end with the X Register Group to the extent that the Registers within these groups are designated by the rightmost 16-bits of Register X0 Right. A positive offset, applied to the PVA designating the first word location of the central memory field, shall begin with zero and shall be increased by eight for each designated transfer as it is accomplished during the course of instruction execution.

The relationship between the bits contained in positions 48 through 63 of Register X0 Right, the 16 registers contained in each of the general register groups A and X, and the positive offset values applied to the beginning address of the word field in central memory, are illustrated in Figure 2.2-1 for the case in which all 32 Registers are transferred.

The leftmost 16-bits of the word locations in the central memory field, which are associated with A Registers to the extent designated, shall not be used by the instruction which loads

NCR/CDC PRIVATE

multiple registers and shall be cleared by the instruction which stores multiple registers.

Special Load A Conditions: The instruction which loads A Registers shall unconditionally transfer only the rightmost 44-bit positions 20 through 63 of each appropriate word from central memory to the corresponding bit positions of the designated A Registers.

When the Load Multiple instruction is executed in virtual addressing mode, the larger value of 1) the 4 bits in bit positions 16 through 19 of each appropriate word from central memory, 2) the leftmost 4 bits in bit positions 16 through 19 of the Aj Register, and 3) the R1 field contained in the 4-bit positions 08 through 11 of the segment descriptor associated with the PVA obtained from Register Aj, shall be transferred to bit positions 16 through 19 of each of the appropriately designated A Registers.

NCR/CDC PRIVATE

When the Load Multiple instruction is executed in real addressing mode, the larger value of, 1) the 4 bits in positions 16 through 19 of each appropriate word from central memory, 2) the leftmost 4 bits of the Aj Register, shall be transferred to bit positions 16 through 19 of each of the appropriately designated A Registers.

With respect to the designated A Registers, when all 4 bits in positions 16 through 19 of any associated word from central memory are equal to zero, a Ring Number Zero condition shall be detected and, following the completion of the Load Multiple instruction's execution, the associated program interruption shall occur.

Notes: For both of these operation codes unless the PVA initially contained in the Aj Register consists of a byte address which is equal to 0, modulo 8, an Address Specification error shall be detected, all transfers associated with the execution of these instructions shall be inhibited, and the corresponding program interruption shall occur.

NCR/CDC PRIVATE

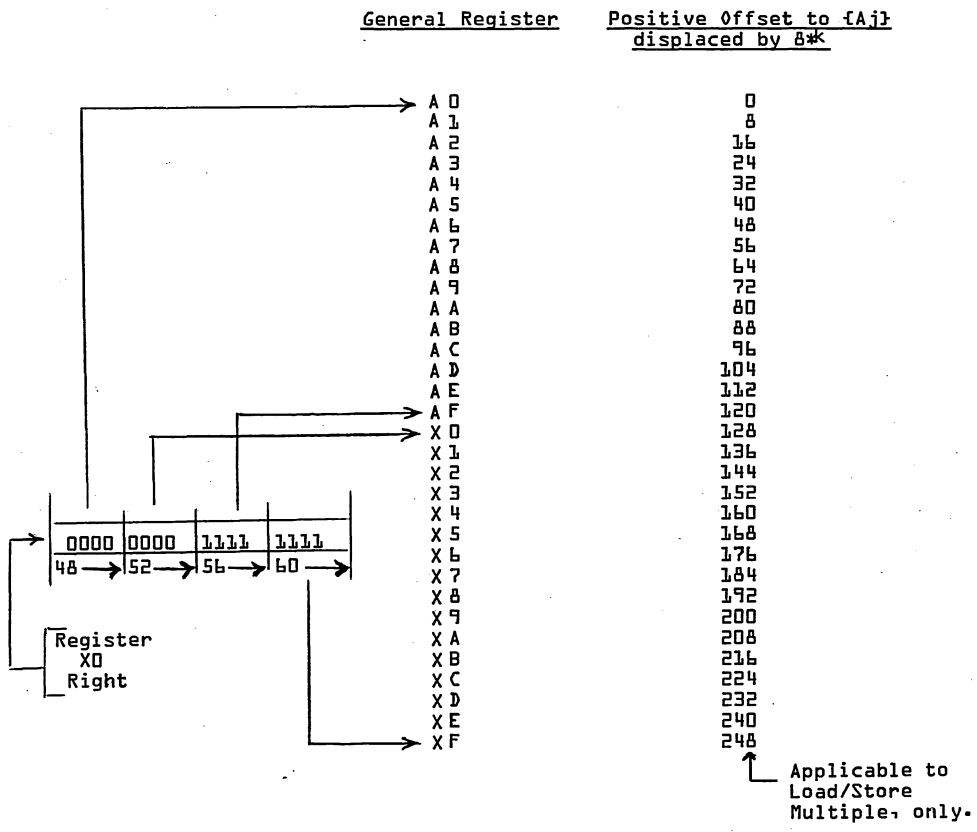


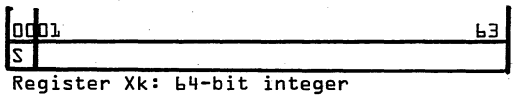
Figure 2.2-1
Register Selectivity Correspondence

NCR/CDC PRIVATE

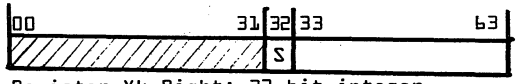
2.2.2 Integer Arithmetic

Integer arithmetic operations shall be performed on words and halfwords contained in Register Xk and Register Xk Right, respectively, as described in the following subparagraphs.

Binary integers contained in the X Registers shall consist of signed, two's complement, 32-bit or 64-bit quantities. The leftmost bit, {in position 00 for 64-bit integers and in position 32 for 32-bit integers}, shall constitute the sign bit. Positive quantities shall consist of a sign bit in the zero state with the 31 or 63 contiguous bits immediately to the right of the sign bit, expressing the magnitude of the number. Negative quantities shall be expressed as the two's complement of their positive representations, resulting in a sign bit in the one state. Conceptually, the two's complement of a binary integer shall be formed by adding one to its one's complement representation. {Conceptually, the one's complement of a binary integer shall be formed by subtracting it, bit-for-bit, from another number consisting entirely of one bits}.



Register Xk: 64-bit integer



Register Xk Right: 32-bit integer

The ranges in magnitude, M, covered by binary integers in each of the two fixed point formats, shall be as follows:

32-bit Integer: $-2^{31} \leq M \leq 2^{31}-1$ 64-bit Integer: $-2^{63} \leq M \leq 2^{63}-1$

2.2.2.1 Integer Sum, Xk

a. Integer Sum, {Xk} replaced by {Xk} plus {Xj}
022 jk

b. Integer Sum, {Xk} replaced by {Xj} plus 0
143 jk0

These instructions shall obtain a 64-bit addend from the initial contents of Register Xj, or from the 16-bit sign extended 0 field of the instruction, as determined by the operation code. The 64-bit addend thus derived shall be added to the 64-bit word initially contained in Register Xk or Xj,

NCR/CDC PRIVATE

as correspondingly determined by the operation code, and shall transfer the 64-bit sum to Register Xk. Each 64-bit word shall be treated as a signed two's complement integer.

When the augend and addend are identically signed, and their addition produces a sum with a sign opposite that of the addend and augend, an Overflow condition shall be detected, and when enabled the corresponding program interruption shall occur.

2.2.2.2 Integer Difference, Xk

Integer Difference, {Xk} replaced by {Xk} minus {Xj}
023 jk

This instruction shall subtract the 64-bit word initially contained in Register Xj from the 64-bit word initially contained in Register Xk and shall transfer the 64-bit difference to Register Xk. Each 64-bit word shall be treated as a signed two's complement integer.

When the minuend and subtrahend are oppositely signed and the subtraction produces a difference with a sign opposite that of the minuend, an Overflow condition shall be detected, and when enabled, the corresponding program interruption shall occur.

2.2.2.3 Integer Product, Xk

Integer Product, {Xk} replaced by {Xk} times {Xj}
024 jk

This instruction shall multiply the 64-bit word initially contained in Register Xk by the 64-bit word initially contained in Register Xj, with each of these 64-bit words treated as signed, two's complement integers. The result of this multiplication shall consist of a 128-bit intermediate product, algebraically signed. The rightmost 64 bits of this intermediate product shall be transferred to the Xk Register.

Unless the leftmost 65 bits of the properly signed intermediate product are all in the same state, an Overflow condition shall be detected and when enabled, the corresponding program interruption shall occur.

NCR/CDC PRIVATE

2.2.2.4 Integer Quotient, Xk

Integer Quotient {Xk} replaced by {Xk} divided by {Xj}

025 jk

This instruction shall divide the 64-bit word initially contained in the Xk Register by the 64-bit word initially contained in the Xj Register. Provided the divisor is not equal to zero, the results of the division, consisting of a 64-bit quotient, algebraically signed, shall be transferred to Register Xk.

When the divisor is equal to zero, the contents of Register Xk shall not be changed, a Divide Fault condition shall be detected, and when enabled, the corresponding program interruption shall occur.

For the case in which -2^{63} is divided by -2^{0}, the quotient result shall have the form of -2^{63}, an Overflow condition shall be detected, and when enabled, the corresponding program interruption shall occur.

Note: The division shall produce a quotient result which, in its absolute form, shall not have been rounded upwards. Thus, when the absolute value of the quotient result is concatenated to a single zero bit, that quantity shall be equal to or less than the absolute value of the quotient computed to one additional bit of precision in the rightmost position. Moreover, when the absolute value of the quotient result is increased by one and concatenated to a single zero bit, that quantity shall be greater than the absolute value of the quotient computed to one additional bit of precision in the rightmost position.

2.2.2.5 Absolute, Xk

Integer {Xk} replaced by Absolute {Xj}

026 jk

This instruction shall treat the 64-bit word initially contained in the Xj Register as a signed two's complement integer and shall transfer its absolute value to the Xk Register. Thus, positively signed values initially contained in Register Xj shall be transferred to Register Xk without change. However, negatively signed

NCR/CDC PRIVATE

values initially contained in Register Xj shall be two's complemented and transferred to Register Xk.

When the 64-bit word transferred to Register Xk maintains a negative sign despite the two's complement operation, an Overflow condition shall be detected and when enabled, the corresponding program interruption shall occur; i.e., $\{Xj\}=80\rightarrow 0_{16}$

2.2.2.6 Integer Sum, Xk Right

a. Integer Sum, {Xk} Right replaced by {Xk} Right plus {Xj} Right
027 jk

b. Integer Sum, {Xk} Right replaced by {Xj} Right plus ℓ
028 jk ℓ

c. Integer Sum, {Xk} Right replaced by {Xk} Right plus j
029 jk

Operation: These instructions shall obtain a 32-bit addend from the initial contents of Register Xj Right, from the 16-bit sign extended ℓ field of the instruction, or from the 4-bit zeros extended j field of the instruction, as determined by the operation code.

The 32-bit addend thus derived, shall be added to the 32-bit halfword initially contained in Register Xk Right or Register Xj Right, as determined by the operation code and the sum shall be transferred to Register Xk Right. Each of these 32-bit halfwords shall be treated as signed two's complement integers.

When the augend and addend are identically signed, and their addition produces a sum with a sign opposite that of the addend and augend, an Overflow condition shall be detected, and when enabled the corresponding program interruption shall occur.

2.2.2.7 Integer Difference, Xk Right

a. Integer Difference, {Xk} Right replaced by {Xk} Right minus {Xj} Right
030 jk

b. Integer Difference, {Xk} Right replaced by {Xk} Right minus j
031 jk

Operation: These instructions shall obtain a 32 bit subtrahend from the initial contents of Register Xj Right or from the 4-bit zeros extended j field from the instruction, as determined by the operation code.

NCR/CDC PRIVATE

The 32-bit subtrahend thus derived shall be subtracted from the 32-bit halfword initially contained in Register Xk Right and the difference shall be transferred to Register Xk Right. Each of these 32-bit halfwords shall be treated as signed two's complement integers.

When the minuend and subtrahend are oppositely signed and the subtraction produces a difference with a sign opposite that of the minuend, an Overflow condition shall be detected, and when enabled the corresponding program interruption shall occur.

2.2.2.8 Integer Product, Xk Right

a. Integer Product, (Xk) Right replaced by (Xk) Right times (Xj) Right
032jk

b. Integer Product, (Xk) Right replaced by (Xj) Right times a
033jka

These instructions shall obtain a 32-bit multiplier from the initial contents of Register Xj Right or from the 16-bit sign extended a field of the instruction, as determined by the operation code.

The 32-bit multiplier thus derived shall be taken times the 32-bit half-word initially contained in Register Xk Right or Register Xj Right as determined by the operation code. The result of the multiplication shall consist of a 64-bit intermediate product, algebraically signed.

The rightmost 32 bits of this intermediate product shall be transferred to Register Xk Right.

Unless the leftmost 33 bits of the properly signed intermediate product are all in the same state, an Overflow condition shall be detected and when enabled, the corresponding program interruption shall occur.

2.2.2.9 Integer Quotient, Xk Right

Integer Quotient, (Xk) Right replaced by (Xk) Right divided by (Xj) Right
034 jk

This instruction shall divide the 32-bit halfword initially

NCR/CDC PRIVATE

contained in Register Xk Right by the 32-bit halfword initially contained in Register Xj Right. Provided the divisor is not equal to zero, the results of the division, consisting of a 32-bit quotient, algebraically signed, shall be transferred to Register Xk Right.

When the divisor is equal to zero, the contents of Register Xk shall not be changed, a Divide Fault condition shall be detected, and when enabled, the corresponding program interruption shall occur.

For the case in which -2^{31} is divided by -2^{0}, the quotient result shall have the form of -2^{31}, an Overflow condition shall be detected, and when enabled, the corresponding program interruption shall occur.

Note: The division shall produce a quotient result which, in its absolute form, shall not have been rounded upwards. Thus, when the absolute value of the quotient result is concatenated to a single zero bit, that quantity shall be equal to or less than the absolute value of the quotient computed to one additional bit of precision in the rightmost position. Moreover, when the absolute value of the quotient result is increased by one and concatenated to a single zero bit, that quantity shall be greater than the absolute value of the quotient computed to one additional bit of precision in the rightmost position.

2.2.2.10 Integer Compare

a. Integer Compare, (Xj) to (Xk), result to X1 Right
035 jk

b. Integer Compare, (Xj) Right to (Xk) Right, result to X1 Right
036 jk

Operation: These instructions shall perform an algebraic comparison of the signed, two's complement, binary integer initially contained in Register Xj to the signed, two's complement, binary integer initially contained in Register Xk. These compared values shall consist of 64-bits or 32-bits (right-justified in positions 32 through 63) as determined by the operation code. In this context the contents of the X0 Register shall be interpreted as consisting entirely of zeros.

Results: When the comparison finds these quantities equal, Register X1 Right shall be cleared in all 32 bit positions. When the comparison finds the quantity obtained from Register Xj greater than the quantity obtained from Register Xk, Register X1

NCR/CDC PRIVATE

Right shall be cleared in the leftmost 31 bit positions, 32 through 62, and shall be set in the rightmost bit position, 63. When the comparison finds the quantity obtained from Register Xj less than the quantity obtained from Register Xk, Register X1 Right shall be set in all 32 bit positions.

2.2.3 Branch

The instructions within this subgroup shall consist of both conditional and unconditional branch instructions.

Each conditional branch instruction shall perform a comparison between the contents of two general registers. Then, based on the relationship between the results of that comparison and the branch condition as specified by means of the instruction's operation code, each conditional branch instruction shall perform either a normal exit or a branch exit.

Normal exit: When the results of a comparison do not satisfy the branch condition as specified by the operation code, a normal exit shall be performed. A normal exit for all conditional branch instructions shall consist of adding four to the rightmost 32 bits of the PVA obtained from the P Register with that 32-bit sum returned to the P Register in its rightmost 32-bit positions.

Branch exit: When the results of a comparison satisfy the branch condition as specified by the operation code, a branch exit shall be performed. A branch exit shall consist of expanding the 16-bit Q field from the instruction to 31 bits by means of sign extension, shifting these 31 bits left one bit position with a zero inserted on the right, and adding this 32-bit shifted result to the rightmost 32-bits of the PVA obtained from the P Register with the 32-bit sum returned to the P Register in its rightmost 32-bit positions.

Unconditional branch instructions shall perform branch exits according to the appropriate instruction descriptions contained in subparagraphs 2.2.3.5 through 2.2.3.6 of this specification.

2.2.3.1 Conditional, X

a. Branch to {P} displaced by 2*Q if {Xj} equal to {Xk}
   037 jkQ

b. Branch to {P} displaced by 2*Q if {Xj} not equal to {Xk}
   038 jkQ

c. Branch to {P} displaced by 2*Q if {Xj} greater than {Xk}
   039 jkQ

d. Branch to {P} displaced by 2*Q if {Xj} not less than {Xk}
   040 jkQ

Each of these instructions shall perform an algebraic comparison of the 64-bit word obtained from Register Xj to the 64-bit word obtained from Register Xk. Each of these 64-bit words shall be treated as signed, two's complement, binary integers. The contents of Register X0 shall be interpreted as consisting entirely of zeros.

These instructions shall perform a normal exit or a branch exit in the manner previously described in Paragraph 2.2.3 of this specification.

2.2.3.2 Conditional, X Right

a. Branch to {P} displaced by 2*Q if {Xj} Right equal to {Xk} Right
   041 jkQ

b. Branch to {P} displaced by 2*Q if {Xj} Right not equal to {Xk} Right
   042 jkQ

c. Branch to {P} displaced by 2*Q if {Xj} Right greater than {Xk} Right
   043 jkQ

d. Branch to {P} displaced by 2*Q if {Xj} Right not less than {Xk} Right
   044 jkQ

Each of these instructions shall perform an algebraic comparison of the 32-bit halfword obtained from Register Xj Right with the 32-bit halfword obtained from Register Xk Right. Each of these 32-bit halfwords shall be treated as signed, two's complement, binary integers. The contents of Register X0 shall be interpreted as consisting entirely of zeros.

These instructions shall perform a normal exit or a branch exit in the manner previously described in Paragraph 2.2.3 of this specification.

2.2.3.3 Conditional, with Increment

Branch to {P} displaced by 2*@ and increment {Xj} if {Xj} less than {Xk}
045 jk@

This instruction shall perform an algebraic comparison of the 64-bit word initially contained in Register Xj with the 64-bit word initially contained in Register Xk. Each of these 64-bit words shall be treated as signed, two's complement, binary integers.

When this comparison does not find the value initially contained in Register Xj less than the value initially contained in Register Xk a normal exit shall be performed in the manner previously described in Paragraph 2.2.3 of this specification.

When this comparison finds the value initially contained in Register Xj less than the value initially contained in Register Xk, a branch exit shall be performed in the manner previously described in Paragraph 2.2.3 of this specification. In addition, the 64-bit word initially contained in Register Xj shall be increased by one in value with the 64-bit result returned to Register Xj.

2.2.3.4 Conditional, Ak

Branch to {P} displaced by 2*@ if SEG{Aj} not equal to SEG{Ak}; else Compare BN{Aj} to BN{Ak}, result to X1 Right.
046 jk@

This instruction shall perform a bit-for-bit comparison between the 12-bit SEG field contained in bit positions 20 through 31 of Register Aj and the 12-bit SEG field contained in bit positions 20 through 31 of Register Ak. When the comparison finds the SEG fields not equal, this instruction shall perform a branch exit in the manner described in paragraph 2.2.3 of this specification.

When the comparison finds the SEG fields equal, this instruction shall perform an algebraic comparison of the 32-bit BN field contained in bit positions 32 through 63 of Register Aj to the 32-bit BN field contained in bit positions 32 through 63 of Register Ak and shall perform a normal exit in the manner described in Paragraph 2.2.3 of this specification.

The algebraic comparison of the BN fields shall treat each of these 32-bit quantities as signed two's complement binary integers

NCR/CDC PRIVATE

and shall store the result of their comparison into Register X1 Right as follows: When the BN fields are equal, Register X1 Right shall be cleared in all 32-bit positions.

When the BN field from Register Aj is greater than the BN field from Register Ak, Register X1 Right shall be cleared in the leftmost 31-bit positions, 32 through 62, and shall be set in the rightmost bit position, 63. When the BN field from Register Aj is less than the BN field from Register Ak, Register X1 Right shall be set in all 32-bit positions.

2.2.3.5 Unconditional Branch, {P} indexed

Branch to {P} indexed by 2*{Xk} Right
047 jk

This instruction shall perform an unconditional branch exit by modifying the contents of the P Register in its rightmost 32-bit positions as follows:

The 32-bit halfword obtained from Register Xk Right shall be shifted left one bit position, end-off with a zero inserted on the right, and the 32-bit shifted result shall be added to the rightmost 32-bits initially contained in the P Register. This 32-bit sum shall be returned to the P Register in its rightmost 32-bit positions.

NCR/CDC PRIVATE

2.2.3.6 Unconditional Branch, {A} indexed

Branch to {Aj} indexed by 2*{Xk} Right
048 jk

In the absence of all associated Virtual Addressing Mechanism exceptions {other than a Page Table Search Without Find condition at the branch address} this instruction shall perform a branch exit by modifying the GK, LK, SEG and BN fields contained in the P Register as follows:

The 12-bit Segment field, SEG, contained in bit positions 20 through 31 of Register Aj shall be transferred to the corresponding 12-bit positions of the P Register.

The 32-bit halfword obtained from Register Xk Right shall be shifted left one bit position, end-off with a zero inserted on the right, and the 32-bit shifted result shall be added to the rightmost 32-bits obtained from Register Aj in bit positions 32 through 63. {In this context, the contents of Register X0 shall be interpreted as consisting entirely of zeroes}. This 32-bit sum shall be transferred to the rightmost 32-bit positions, 32 through 63, of the P Register.

The Global Key field initially contained in the P Register shall be checked and conditionally altered, and the Local Key field initially contained in the P Register shall be altered, according to the descriptions contained in subparagraph 3.6.3.2 of this specification.

The P{RN} field shall not be changed by the execution of this instruction. Moreover, the Execute validation procedure for the next instruction fetch; i.e. the fetching of the instruction at the branch address, shall be included in this branch instruction's execution such that the detection of associated Access Violations, as described in Subparagraphs 3.3.1.1 and 3.6.2.1 of this specification, shall result in the corresponding program interruption and the execution of this instruction shall be inhibited.

Note: Unless the PVA contained in Register Aj consists of a byte address which is equal to 0, modulo 2, an Address Specification error shall be detected, the execution of this instruction shall be inhibited and the corresponding program interruption shall occur.

2.2.4 Copy

The instructions within this subgroup shall provide the means for accomplishing inter-register transfers to the extent defined by the following instruction descriptions.

2.2.4.1 Copy to Xk from Xj
049 jk

This instruction shall transfer the 64-bit word initially contained in Register Xj to the 64-bit positions of Register Xk.

2.2.4.2 Copy to Xk from Aj
050 jk

This instruction shall transfer the 48 bits contained in Register Aj to the rightmost 48-bit positions, 16 through 63, of Register Xk. The leftmost 16-bit positions, 00 through 15, of Register Xk shall be cleared.

2.2.4.3 Copy to Ak from Aj
051 jk

This instruction shall transfer the 48 bits contained in Register Aj to the 48-bit positions of Register Ak.

2.2.4.4 Copy to Ak from Xj
052 jk

This instruction shall unconditionally transfer the rightmost 44 bits, contained in positions 20 through 63, of Register Xj to the corresponding 44-bit positions of Register Ak. The 4-bit field having the larger value in bit positions 16 through 19 of the Xj Register or the P Register, shall be transferred to the corresponding 4-bit positions of the Ak Register.

2.2.4.5 Copy to Xk Right from Xj Right
053 jk

This instruction shall transfer the 32-bit halfword initially contained in Register Xj Right to the 32-bit positions, 32 through 63, of Register Xk Right. The initial contents of Register Xk Left shall not be changed.

2.2.5 Address Arithmetic

The instructions within this subgroup shall provide the means

for accomplishing address arithmetic to the extent defined by the following instruction descriptions.

2.2.5.1 Copy A with Displacement

Address {Ak} replaced by {Aj} plus Q
054 jkQ

This instruction shall transfer the leftmost 16 bits initially contained in bit positions 16 through 31 of Register Aj to the corresponding 16-bit positions of Register Ak. In addition, the 16-bit Q field from the instruction, expanded to 32-bits by means of sign extension, shall be added to the rightmost 32 bits initially contained in bit positions 32 through 63 of Register Aj and the 32-bit sum shall be transferred to the corresponding rightmost 32-bit positions of Register Ak.

2.2.5.2 | Copy P with Indexing and Displacement

| Address {Ak} replaced by {P} plus 2* {Xj} Right plus 2*Q
055 jkQ

This instruction shall transfer the leftmost 16 bits contained in bit positions 16 through 31 of the P Register to the corresponding 16-bit positions of the Ak Register. In addition, the 16-bit Q field from the instruction shall be expanded to 31 bits by means of sign extension, these 31 bits shall be shifted left one bit position with a zero inserted on the right, and this 32-bit shifted result shall be added to the rightmost 32 bits obtained from the P Register. This 32-bit sum shall be added to the rightmost 32-bits obtained from Register Xj Right, shifted left one bit position with a zero inserted on the right, and the final result shall be transferred to the rightmost 32-bit positions, 32 through 63, of Register Ak. In this context, the contents of Register X0 shall be interpreted as consisting entirely of zeros.

2.2.5.3 A Indexed

Address {Ak} replaced by {Ak} plus {Xj} Right
056 jk

This instruction shall add the 32-bits contained in Register Xj Right to the rightmost 32-bits initially contained in bit positions 32 through 63 of Register Ak and shall return the 32-bit sum to the rightmost 32-bit positions of Register Ak.

2.2.6 Enter

The instructions within this subgroup shall provide the means

NCR/CDC PRIVATE

for entering immediate operands, {consisting of logical quantities or signed, two's complement binary integers}, into the X Registers to the extent defined by the following instruction descriptions.

2.2.6.1 Enter j

a. Enter Xk Right with plus j
057 jk

b. Enter Xk Right with minus j
058 jk

Operation. These instructions shall expand the 4-bit j field from the instruction to 32-bits by extending 28 zeros on the left and shall transfer this 32-bit result or the two's complement of this 32-bit result, as determined by the operation code, to the 32-bit positions, 32 through 63, of Register Xk Right.

2.2.6.2 Enter Q

Enter Xk Right with sign extended Q
059 jkQ

This instruction shall expand the 16-bit Q field from the instruction to 32-bits by means of sign extension and shall transfer this 32-bit result to the 32-bit positions, 32 through 63, of Register Xk Right.

2.2.6.3 Enter jk

Enter X0 Right with logical jk
060 jk

This instruction shall transfer the 4-bit k field from the instruction to bit positions 60 through 63 of Register X0 Right, shall transfer the 4-bit j field from the instruction to bit positions 56 through 59 of Register X0 Right, and shall clear the leftmost 24 bit positions, 32 through 55 of Register X0 Right.

2.2.6.4 Enter signs

Enter Xk Left with signs per j
061 jk

| The value of the rightmost 2-bits of the j field from the instruction shall be translated as follows:

NCR/CDC PRIVATE

b. When the 4-bit j field from this instruction is equal to 1, the 32-bit positions 00 through 31, of Register Xk Left shall be set.

c. When the 4-bit j field from this instruction is equal to 2, the sign bit in position 32 of Register Xk Right shall be transferred to all 32-bit positions, 00 through 31 of Register Xk Left.

d. When the 4-bit j field from this instruction is equal to 3 through F (hex), the execution of this instruction shall result in no operation.

2.2.7 Shift

The instructions within this subgroup shall provide the means for shifting the initial contents of the Xj Register and transferring the result to the Xk Register, to the extent defined by the following instructions.

All of the instructions within this subgroup shall derive the computed shift count in the following manner: The 12-bit D field from these instructions shall be expanded to 32-bits by means of sign extension, these 32-bits shall be added to the 32-bits initially obtained from Register Xi Right and this 32-bit sum shall represent the computed shift count. In this context the contents of Register X0 Right shall be interpreted as consisting entirely of zeros.

The instructions within this subgroup shall interpret the computed shift count as follows: The sign bit in the leftmost position of the 32-bit computed shift count shall determine the direction of the shift. When the computed shift count is positive, (sign bit of zero), these instructions shall left shift, with the number of bit positions to be shifted determined by the value of the computed shift count in its rightmost 5-bit and 6-bit positions for 32-bit and 64-bit operands, respectively. When the computed shift count is negative, (sign bit of one), these instructions shall right shift, with the number of bit positions to be shifted determined by the two's complement of the computed shift count in its rightmost 5-bit and 6-bit positions for 32-bit and 64-bit operands, respectively.

When these interpretations of the computed shift count result in an actual shift count of zero, the associated instructions shall transfer the initial contents of the Xj Register to the Xk Register and no shifting shall be performed.

2.2.7.1 Shift (Xj) to Xk, Circular

Shift (Xj) to Xk Circular, Direction and Count per (Xi) Right plus D
062 jkiD

NCR/CDC PRIVATE

This instruction shall shift the 64-bit word initially contained in Register Xj, with the direction and number of bit positions to be shifted determined by the computed shift count, and shall transfer the result to Register Xk. The computed shift count shall be derived and interpreted in the manner described in Paragraph 2.2.7 of this specification.

This instruction shall shift circularly such that bits shifted out one end of the 64-bit word shall be transferred into bit positions which become unoccupied at the opposite end of the 64-bit word as a result of the shift.

2.2.7.2 Shift (Xj) to Xk, End-off

a. Shift (Xj) to Xk, Direction and Count per (Xi) Right plus D
063 jkiD

b. Shift (Xj) Right to Xk Right, Direction and Count per (Xi) Right plus D
064 jkiD

Operation: These instructions shall shift the 64-bit word initially contained in Register Xj or the 32-bit half word contained in Register Xj Right, as determined by the operation code, and shall transfer the result to Register Xk or Register Xk Right as correspondingly determined by the operation code. The direction and number of bit positions to be shifted shall be determined by the computed shift count. The computed shift count shall be derived and interpreted in the manner described in Paragraph 2.2.7 of this specification.

Right Shift: Right shifts shall be performed end-off on the right and sign extended on the left. Thus, bits shifted out of the rightmost bit position shall be lost and the leftmost bit position, which would otherwise become unoccupied for each bit position shifted, shall be left unchanged.

Left Shift: Left shifts shall be performed end-off on the left with zeros inserted on the right. Thus, bits shifted out of the leftmost bit position shall be lost and the rightmost bit position, which becomes unoccupied for each bit position shifted, shall be cleared.

2.2.8 Logical

The instructions within this subgroup shall provide the means for performing Boolean operations on the 64-bit words contained in the X Registers to the extent defined by the following instruction descriptions.

NCR/CDC PRIVATE

2.2.8.1 Logical Sum, Difference, and Product

a. Logical Sum, (Xk) replaced by (Xk) OR (Xj)
   065 jk

b. Logical Difference, (Xk) replaced by (Xk) EOR (Xj)
   066 jk

c. Logical Product, (Xk) replaced by (Xk) AND (Xj)
   067 jk

These instructions shall perform a logical operation between the 64-bit word initially contained in the Xj Register and the 64-bit word initially contained in the Xk Register and shall return the 64-bit Boolean result to the Xk Register.

The logical operations performed by these instructions shall consist of a logical sum (OR), a logical difference (EOR) or a logical product (AND), as determined by the operation code, and accomplished according to the following truth tables.

```
OR:  0011      EOR:  0011      AND:  0011
     0101            0101            0101
     ----            ----            ----
     0111            0110            0001
```

2.2.8.2 Logical Complement

Logical Complement, (Xk) replaced by (Xj) NOT
068 jk

This instruction shall transfer the one's complement of the 64-bit word initially contained in the Xj Register to the 64-bit positions of the Xk Register.

Conceptually, taking the one's complement of a 64-bit word shall be accomplished by subtracting it, bit-for-bit, from a 64-bit word consisting entirely of one bits.

```
One's Complement Truth Table:  1's   :  1111
                               (Xj)  :  0110
                               (Xk)  :  1001
```

2.2.8.3 Logical Inhibit

Logical Inhibit, (Xk) replaced by (Xk) AND (Xj) NOT
069 jk

NCR/CDC PRIVATE

This instruction shall perform a logical product between the one's complement of the 64-bit word initially contained in the Xj register and the 64-bit word initially contained in the Xk register and shall return the 64-bit Boolean result to the Xk register.

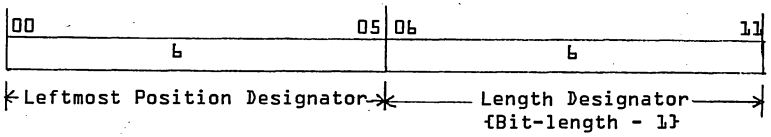
The truth tables for the logical product and one's complement operations are provided in Subparagraphs 2.2.8.1 and 2.2.8.2, respectively, of this specification.

2.2.9 Register Bit String

The instructions within this subgroup shall provide the means for addressing a contiguous string (field) of bits, beginning and ending independently with any bit positions within a 64-bit word.

For each of the instructions in this subgroup, the bit strings shall be addressed by means of a 12-bit field referred to as a bit string descriptor. Any field of bits, including the field constituting a bit field descriptor, shall be numbered from left to right, with the leftmost bit numbered 00. The six-bit subfield in bit positions 00 through 05 of a bit string descriptor shall designate the beginning, or leftmost, bit position within a 64-bit word. The 6-bit subfield in bit positions 06 through 11 of the bit string descriptor is a length designator that is interpreted as designating one less than the length (in bits) of a bit string within a 64-bit word.

Bit String Descriptor



For all instructions within this subgroup, indexing shall be carried out as follows: the bit string descriptor obtained from the D field of the instruction shall be zero-extended on the left to 32 bits and then added, without overflow detection, to the contents of register Xi Right (in this context, the contents of register X0 shall be interpreted as all zeroes); the rightmost 12 bits of the result shall then be interpreted as a bit string descriptor, in the manner described above. For each of the instructions in this subgroup, when, after indexing, the sum of the "Leftmost Position Designator" and the "Length Designator" is greater than 63 (decimal), an Instruction Specification error shall be detected, the execution of the associated instruction shall be inhibited and the corresponding program interruption shall occur.

NCR/CDC PRIVATE

2.2.9.1 Isolate Bit Mask

Isolate Bit Mask into Xk per (Xi) Right plus D
070 jkiD

This instruction shall generate, in Xk, a bit mask consisting of a field of contiguous one bits whose leftmost and rightmost bit positions are determined by the bit field descriptor calculated and interpreted as specified in subparagraph 2.2.9. All bit positions to the left of the leftmost bit position and all bit positions to the right of the rightmost bit position (leftmost bit position plus length designator), if any, shall consist of zeroes.

2.2.9.2 Isolate to Xk

Isolate into Xk from Xj per (Xi) Right plus D
071 jkiD

This instruction shall obtain a field of contiguous bits from the initial contents of the Xj register, shall clear all 64 bit positions of the Xk register, and shall then transfer that field of contiguous bits, right justified, into the Xk register. The leftmost and rightmost bit positions of the field obtained from the Xj register shall be defined by the bit field descriptor calculated and interpreted as specified in subparagraph 2.2.9.

2.2.9.3 Insert into Xk

Insert into Xk from Xj per (Xi) Right plus D
072 jkiD

This instruction shall transfer a field of contiguous bits, initially contained right justified in the Xj register, to a field of contiguous bit positions in the Xk register. The length of the bit string obtained from the Xj register, and the leftmost and rightmost bit positions of the Xk register shall be defined by the bit string descriptor calculated and interpreted as specified in paragraph 2.2.9. All bit positions to the left of the leftmost bit position, and all bit positions to the right of the rightmost bit position of the Xk register, if any, shall be left unchanged.

NCR/CDC PRIVATE

2.2.10 Move, Compare and Logical

The instructions in this sub-group shall provide the means for accomplishing memory-to-memory move, comparison and logical operations on variable-length byte fields, referred to as source and destination fields.

The leftmost byte addresses of the source and destination fields shall be designated by the PVA's contained in Registers Aj and Ak, respectively. The lengths of the source and destination fields shall be designated by the 32-bit halfwords contained in Registers X0 Right and X1 Right, respectively.

The source and destination fields shall be processed from left to right. For the reading of each byte associated with the source field, the BN field contained in Register Aj shall be incremented by one and the 32-bit halfword contained in Register X0 Right shall be decremented by one. For the processing, reading and/or writing, of each byte associated with the destination field, the BN field contained in Register Ak shall be incremented by one and the 32-bit halfword contained in Register X1 Right shall be decremented by one.

Whenever either field has been exhausted, i.e., its length has been decremented to zero, increment and decrement operations on the contents of its associated A and X Registers, respectively, shall be inhibited.

For the purpose of establishing operand access validation in virtual addressing mode, each central memory operand access type shall be a read access with respect to bytes read from the source and destination fields and a write access with respect to bytes stored into the destination field.

For each of these instructions, when the initial contents of Register X0 Right or X1 Right are negative (bit 32 equal to a one), an Instruction Specification error shall be recorded, the execution of the associated instruction shall be inhibited, and the corresponding program interruption shall occur.

a. Move

Move Bytes Direct, (Ak) replaced by (Aj) per X0 and X1.
073jk

Move and Complement Bytes Direct, (Ak) replaced by $\overline{(Aj)}$ per X0 and X1.
153jk

These instructions shall move from 0 to 256 bytes from a source field in central memory to a destination field in central memory.

NCR/CDC PRIVATE

Fill uncomplemented fore.

Each byte from the source field shall be moved to the destination field unchanged, or one's complemented, as determined by the operation code. The bit-for-bit one's complement operation is described in the truth table contained in subparagraph 2.2.8.2 of this specification.

When the source field has been exhausted, a fill byte from bit positions 24 through 31 of Register X0 Left shall be used as often as required to complete the Move operation.

When the Move operation terminates as a result of exhausting the destination field, the 32-bit positions, 00 through 31, of Register X1 Left shall be cleared.

When the Move operation terminates as a result of storing a maximum of 256 bytes in the destination field without exhausting the destination field, the 32-bit halfword contained in Register X1 Left shall be set in all 32-bit positions, 00 through 31.

b. Compare Bytes Direct, {Aj} to {Ak} per X0 and X1
144jk

This instruction shall compare a maximum of 256 bytes from the source field in central memory to a maximum of 256 corresponding bytes from the destination field in central memory. The comparison shall occur from left to right, byte-by-byte, with each byte treated as an unsigned, binary value.

When the lengths of the two fields are unequal, a fill-byte from bit positions 24 through 31 of Register X0 Left shall be used, as often as required, to continue the comparison operation after the field having the shorter length has been exhausted.

When the operation terminates as a result of inequality between two corresponding bytes, as individually associated with the source and destination fields, Register X1 Left shall be cleared in all 32 bit positions, 00 through 31, and the results of the unequal comparison shall be transferred to Register X1 Right in the manner described in subparagraph 2.2.2.10 of this specification, {with the source and destination fields analogous to the Xj and Xk Registers, respectively}

When both the source and destination fields are exhausted without the occurrence of inequality, Register X1 Left shall be cleared in all 32-bit positions.

When a maximum of 256 comparisons have been performed, without the occurrence of inequality and without exhausting both field lengths, Register X1 Left shall be set in all 32-bit positions, 00 through 31.

NCR / CDC PRIVATE

When the initial contents of both Register X0 Right and Register X1 Right consist entirely of zeroes, Register X1 Left shall be cleared in all 32-bit positions and no comparisons shall be performed.

c. Logical

Logical Sum, {Ak} replaced by {Ak} OR {Aj} per X0 and X1
151jk

Logical Difference, {Ak} replaced by {Ak} EOR {Aj} per X0 and X1
152jk

Logical Product, {Ak} replaced by {Ak} AND {Aj} per X0 and X1
150jk

Each of these instructions shall perform a logical operation between a maximum of 256 bytes from the source field in central memory and a maximum of 256 corresponding bytes from the destination field in central memory. The logical operation shall consist of a Logical Sum, {OR}, Logical Difference {EOR}, or Logical Product {AND} as determined by the operation code. The bit-for-bit OR, EOR, and AND are defined by the truth tables contained in subparagraph 2.2.8.1 of this specification.

For each of these instructions, when the source field is exhausted, a fill-byte from bit positions 24 through 31 of Register X0 Left, shall be used, as often as required, to complete the logical operation.

When these logical operations terminate as a result of exhausting the destination field, the 32-bit positions 00 through 31, of Register X1 Left shall be cleared.

When these logical operations terminate as a result of storing 256 bytes in the destination field without exhausting the destination field, the 32-bit halfword in Register X1 Left shall be set in all 32-bit positions, 00 through 31.

For each of these instructions, when the source field is to the left of the destination field and the fields overlap, the results of the logical operation shall be undefined.

Novalidity Check.

NCR / CDC PRIVATE

2.2.11 Mark to Boolean

Set Xk Right per j and {X1} Right

145 jk

This instruction shall test the two bits initially contained in the rightmost two bit positions, b2 and b3, of Register X1 Right according to the 4-bit j field from the instruction. When the value of the two rightmost bits initially contained in Register X1 Right is equal to any of the one or more values specified by the instruction's j field, Register Xk Right shall be cleared in bit positions 32 through b2 and shall be set in bit position b3. When the value of the two rightmost bits initially contained in Register X1 Right is not equal to any of the one or more values specified by the instruction's j field, Register Xk Right shall be cleared in all 32 bit positions, 32 through b3. The values of the j field and the rightmost two bits initially contained in Register X1 Right shall be interpreted with respect to equality {EQ} as follows:

j	Bits b2 and b3 of X1 Right, respectively.			
	00	01	10	11
0000	Unconditional inequality			
0001				EQ
0010			EQ	
0011			EQ	EQ
0100		EQ		
0101		EQ		EQ
0110		EQ	EQ	
0111		EQ	EQ	EQ
1000	EQ			
1001	EQ			EQ
1010	EQ		EQ	
1011	EQ		EQ	EQ
1100	EQ	EQ		
1101	EQ	EQ		EQ
1110	EQ	EQ	EQ	
1111	Unconditional Equality			

Note: The four individual bits of j can be visualized as individual pointers which are associated, from left to right, with the four possible values {00, 01, 10 and 11} of the tested bit-pair {bits b2 and b3 of Register X1 Right}. For example, if j = 0101, equality shall be detected when the value of the tested bit pair is 01 or 11.

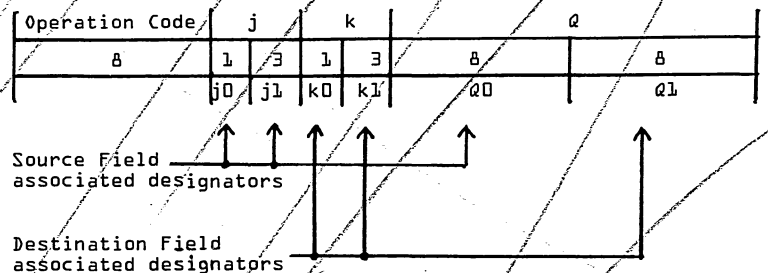
NCR/CDC PRIVATE

2.3 Business Data Processing Instructions

The general form of execution for the instructions in this group shall involve the utilization of a first data field in central memory, referred to as the "source," to modify, replace, or compare with a second data field in central memory referred to as the "destination." Both the source and destination fields shall be individually described by means of independently designated Data Descriptors, with respect to the types of representation, sign and zone conventions, lengths and relative locations of the data fields. Each Data Descriptor, consisting of a 32-bit half-word or a 64-bit word, shall be obtained from an associated table in central memory referred to as a Data Descriptor Table, {DDT}

2.3.1 General Description

The instructions in this group shall be further characterized by their exclusive use of the jkQ/2 instruction format, interpreted as follows:



2.3.1.1 Operation Codes

A total of 31 operation codes shall be utilized by the instructions comprising the BDP Instruction group. These instructions are individually listed with their full names in Appendix A of this specification. For the purpose of this specification, the BDP Instruction group shall be further divided into six subgroups, including "short" instruction names, as follows: _

Note: For the order of exception sensing for these instructions, as well as all other instructions, see paragraph 2.8.7 of this specification.

NCR/CDC PRIVATE

2.2.11 Mark to Boolean

Set Xk Right per j and {X1} Right

145 jk

This instruction shall test the two bits initially contained in the rightmost two bit positions, b2 and b3, of Register X1 Right according to the 4-bit j field from the instruction. When the value of the two rightmost bits initially contained in Register X1 Right is equal to any of the one or more values specified by the instruction's j field, Register Xk Right shall be cleared in bit positions 32 through b2 and shall be set in bit position b3. When the value of the two rightmost bits initially contained in Register X1 Right is not equal to any of the one or more values specified by the instruction's j field, Register Xk Right shall be cleared in all 32 bit positions, 32 through b3. The values of the j field and the rightmost two bits initially contained in Register X1 Right shall be interpreted with respect to equality {EQ} as follows:

j	Bits b2 and b3 of X1 Right, respectively:			
	00	01	10	11
0000	Unconditional inequality			
0001				EQ
0010			EQ	
0011			EQ	EQ
0100		EQ		
0101		EQ		EQ
0110		EQ	EQ	
0111		EQ	EQ	EQ
1000	EQ			
1001	EQ			EQ
1010	EQ		EQ	
1011	EQ		EQ	EQ
1100	EQ	EQ		
1101	EQ	EQ		EQ
1110	EQ	EQ	EQ	
1111	Unconditional Equality			

Note: The four bits of j can be visualized as pointers which are associated {from left to right} with the four possible values {00, 01, 10 and 11} of the tested bit-pair {bits b2 and b3 of Register X1 Right}. For example, if j = 0101, equality shall be detected when the value of the tested bit pair is 01 or 11.

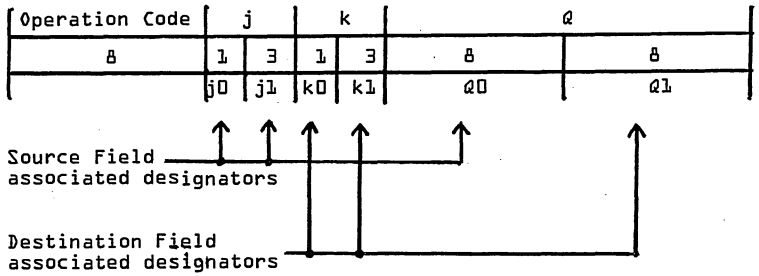
NCR/CDC PRIVATE

2.3 Business Data Processing Instructions

The general form of execution for the instructions in this group shall involve the utilization of a first data field in central memory, referred to as the "source," to modify, replace, or compare with a second data field in central memory referred to as the "destination." Both the source and destination fields shall be individually described by means of independently designated Data Descriptors, with respect to the types of representation, sign and zone conventions, lengths and relative locations of the data fields. Each Data Descriptor, consisting of a 32-bit half-word or a 64-bit word, shall be obtained from an associated table in central memory referred to as a Data Descriptor Table. {DDT}

2.3.1 General Description

The instructions in this group shall be further characterized by their exclusive use of the jkQ/2 instruction format, interpreted as follows:



2.3.1.1 Operation Codes

A total of 31 operation codes shall be utilized by the instructions comprising the BDP Instruction group. These instructions are individually listed with their full names in Appendix A of this specification. For the purpose of this specification, the BDP Instruction group shall be further divided into six subgroups, including "short" instruction names, as follows:

Note: For the order of exception sensing for these instructions, as well as all other instructions, see paragraph 2.8.7 of this specification.

NCR/CDC PRIVATE

Subgroup	Short Name (31)
BDP Numeric	Sum Difference Product Quotient Scale Scale Rounded Decimal Compare Numeric Move
Byte	Compare Compare Collated Scan While Non-Member Translate Move Bytes Edit
Descriptor	Move Table Entry Increment by Table Entry Insert Extract Increment by X-Register Decrement by X-Register Calculate Subscript and Move
Logical	AND OR XOR NOT
Immediate Data	Move Immediate Data Compare Immediate Data Add Immediate Data
Register Load	Load A-Register Load X-Register Store X-Register

? Maybe ?

NCR/CDC PRIVATE

2.3.1.2 Source Field Designators

Designators associated with source fields shall be interpreted from jk@/2 formatted instructions as follows:

The 1-bit j0 field shall be interpreted in the zero state as designating Register A4, and in the one state as designating Register A5, with respect to specifying the A Register which contains the PVA corresponding to the first location in central memory of the Data Descriptor Table associated with the source field.

The 8-bit @0 field shall be interpreted as a halfword item count with respect to designating a specific Data Descriptor in central memory as contained in the DDT associated with the source field. Thus, the 8-bit @0 field shall be expanded to 30 bits by extending zeros on its left, these 30 bits shall be shifted left 2 bit positions with zeros inserted on the right, and the 32-bit shifted result shall be added to the rightmost 32 bits of the PVA obtained from Register A4 or Register A5, (as determined by j0,) with respect to the formation of the central memory word or halfword address corresponding to the location of a specific Data Descriptor associated with the source field.

The 3-bit j1 field, after concatenation to a one bit in the most significant position, shall be interpreted according to the resulting value, 8 through 15, as designating one of the eight A Registers, A8 through AF, respectively, with respect to specifying the A Register which contains the PVA corresponding to the unmodified leftmost byte address (base) of the source field in central memory. **See Figure 2.3-1.**

2.3.1.3 Destination Field Designators

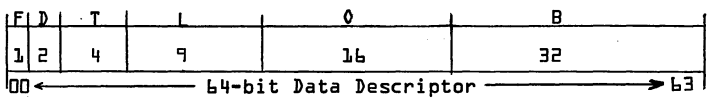
Designators associated with destination fields shall be interpreted from jk@/2 formatted instructions as follows:

The 1-bit k0 field shall be interpreted in the zero state as designating Register A4, and in the one state as designating Register A5, with respect to specifying the A Register which contains the PVA corresponding to the first location in central memory of the Data Descriptor Table associated with the destination field.

NCR/CDC PRIVATE

2.3.2.2 Word DATA DESCRIPTOR.

Data Descriptors where the F field is equal to 1 shall be 64-bits in length, shall be located on any word boundary within a DDT and shall be formatted as follows:



Unless the PVA used to access a 64-bit word as a Data Descriptor is equal to 0, modulo 8, an (Address Specification) error shall be detected, the associated BDP instruction execution shall be inhibited and the corresponding program interruption shall take place.

0-8 (Word Boundry)

Word Data Descriptors shall utilize the 16-bit 0 field, in bit positions 16 through 31 as a byte item count to be applied as a signed offset {2's complement for negative offset} to the leftmost byte address {base} of the associated source or destination field in a manner identical to that previously described for Halfword Data Descriptors. However, Word Data Descriptors shall also utilize the 32-bit B field, in bit positions 32 through 63 as a byte item count to be applied as a signed index {two's complement if negative}, to the leftmost byte address {base} of the associated source or destination field. Thus, both the signed offset represented by the 16-bit 0 field, and the signed index represented by the 32-bit B field, shall be added to the rightmost 32-bit positions of the PVA obtained from the A Register specified by j1+8, or the A Register specified by k1+8, with respect to the formation of the leftmost byte address of the source or destination field, respectively.

NCR/CDC PRIVATE

2.3.2.3 Field, D, T and L

The D, T, and L fields shall be identically interpreted for both Data Descriptor formats, as follows:

a. D Field: The D field is a reserved field consisting of 2 bits in positions 01 and 02 of the Data Descriptor.

NCR/CDC PRIVATE

b. T Field: The T field shall consist of 4 bits, in bit positions 03 through 06 of the Data Descriptor, and shall describe the type of data representation used in the associated source or destination field. The 16 values of the T field are assigned data type representations as follows:

0	Packed Decimal No Sign
1	Packed Decimal No Sign Leading Slack Digit
2	Packed Decimal Signed
3	Packed Decimal Signed Leading Slack Digit
4	Unpacked Decimal Unsigned
5	Unpacked Decimal Trailing Sign Combined Hollerith
6	Unpacked Decimal Trailing Sign Separate
7	Unpacked Decimal Leading Sign Combined Hollerith
8	Unpacked Decimal Leading Sign Separate
9	Alphanumeric
10	Binary Unsigned
11	Binary Signed
12-15	Reserved

As determined by the operation code, source and destination field data types shall be restricted to only those combinations which are defined as valid within the instruction descriptions. The designation of invalid T field combinations within the associated Data Descriptors shall result in the detection of an Instruction Specification error, the instruction's execution shall be inhibited and the corresponding program interruption shall occur. See 2.8.1.4.

NCR/CDC PRIVATE

c. L Field: The L field shall consist of 9 bits, in bit positions 07 through 15, and shall describe the length of the associated source or destination field. The length shall be expressed in terms of bytes. Although the L field can contain larger values (maximum 511), the length of a BDP operand shall be restricted to a smaller value according to the operand data type. These inclusive limits are the following:

19 bytes for Packed Decimal (types 0 through 3);

8X

38 bytes for Unpacked Decimal (types 4 through 8)

8 bytes for Binary (types 10 and 11);

256 bytes for Alphanumeric (type 9).

When any L field exceeds the specified maximum associated with a given data type, an Instruction Specification error shall be detected the execution of that instruction shall be inhibited and the corresponding program interruption shall occur. See 2.8.1.4.

2.3.2.4 Overlap

The execution of BDP Instructions shall be undefined, with respect to the generated results, for every case in which the source and destination fields overlap and are not coincident in their leftmost and rightmost byte positions.

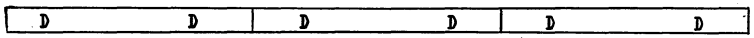
2.3.2.5 Data and Sign Conventions

With respect to numeric data and sign conventions, interpretation shall be performed according to Type (T) where applicable, for characters (C), Digits (D) and Signs (S), using hexadecimal notation, as follows:

Note: Data field examples are illustrated as three byte fields.

NCR/CDC PRIVATE

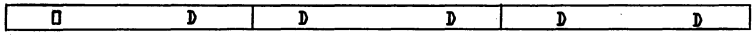
a. Type 0: Packed Decimal No Sign



D: Hex{0} through hex{9}; Decimal 0 through 9, respectively.

Note: This format corresponds to an even number of digits in the decimal number.

b. Type 1: Packed Decimal No Sign Slack Digit



0: Hex{0}; Decimal 0
D: Hex{0} through hex{9}; Decimal 0 through 9, respectively.

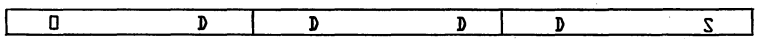
Note: This format corresponds to an odd number of digits in the decimal number.

c. Type 2: Packed Decimal Signed

D: Hex{0} through hex{9}; Decimal 0 through 9, respectively.
S: Hex{A}, {B}, {C}, {E}, or {F} : positive {hex{C} is preferred};
Hex{D} : negative.

Note: This format corresponds to an odd number of digits in the decimal number.

d: Type 3: Packed Decimal Signed Slack Digit

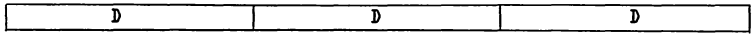


0: Hex{0}; Decimal 0
D: Hex{0} through hex{9}; Decimal 0 through 9, respectively.
S: Hex{A}, {B}, {C}, {E}, or {F} : positive {hex{C} is preferred};
Hex{D} : negative.

Note: This format corresponds to an even number of digits in the decimal number.

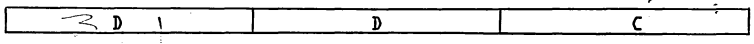
NCR/CDC PRIVATE

e. Type 4: Unpacked Decimal Unsigned



D: ASCII character 0 through 9 represented by hex{30} through hex{39}, respectively.

f. Type 5: Unpacked Decimal Trailing Sign Combined Hollerith



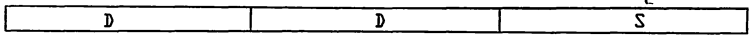
D: ASCII character 0 through 9 represented by hex{30} through hex{39}, respectively;

C: An ASCII character decoded as follows:

ASCII 1 through 9 {hex{31} through hex{39}} ASCII A through I {hex{41} through hex{49}}	either represents +1 through +9
ASCII J through R {hex{4A} through hex{4F} and hex{50} through hex{52}}	represents -1 through -9
ASCII {,0,& {hex{7B}, hex{30}, hex{26}}	represents +0
ASCII },- {hex{7D}, hex{2D}}	represents -0

Note: The underlined characters and codes are the preferred ones.

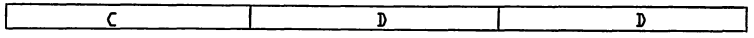
g. Type 6: Unpacked Decimal Trailing Sign Separate



D: ASCII character 0 through 9 represented by hex{30} through hex{39}, respectively.

S: ASCII character + {hex{2B}} : positive sign;
ASCII character - {hex{2D}} : negative sign.

h. Type 7: Unpacked Decimal Leading Sign Combined Hollerith



C and D have the same meaning as for type 5 in subparagraph f.

NCR/CDC PRIVATE

i. Type 8: Unpacked Decimal Leading Sign Separate

S	D	D

D and S have the same meaning as for type 6 in subparagraph g.

j. Type 9: Alphanumeric

C	C	C

C: Any ASCII character code.

k. Type 10: Binary Unsigned

The field defined by the number of bytes contains the positive binary value of the operand.

l. Type 11: Binary Signed

The field defined by the number of bytes contains the signed binary value of the operand, negative values being represented in the 2's complement form.

2.3.3 BDP Numeric

The instructions in this subgroup shall provide the means for performing arithmetic, shift, conversion and comparison operations for byte fields in central memory consisting of numeric decimal data.

Unless the length, type and format fields within the Data Descriptors associated with the source and destination fields, conform to the restrictions defined within the following instruction descriptions, the detection of a Length, Type or Data Descriptor Specification error shall result in an Instruction Specification Error condition, the execution of the associated instruction shall be inhibited and the corresponding program interruption shall occur.

NCR/CDC PRIVATE

When invalid decimal data is encountered in the course of executing one of the following instructions, an Invalid BDP Data condition shall be detected and upon completion of the associated instruction execution, the corresponding program interruption shall occur, if enabled.

When the results from a decimal operation exceed the capacity of the designated destination field such that significant digits are not stored into central memory, [i.e., leftmost non-zero digits are truncated], a Decimal Significance Loss shall be detected and upon completion of the associated instruction's execution, the corresponding program interruption shall occur, if enabled.

Leading zeros shall be supplied and leading digits shall be truncated with respect to accommodating unequal source and destination field lengths. Thus, conceptually, these instructions shall process the data fields from right to left.

Note that these conventions shall cover the end cases for numeric operands of length equal to 1 for all numeric data types. For instance, a Move Numeric from a type 5 operand to a type 3 or type 6 operand of length 1 would amount to an extraction of the source field sign.

A source BDP operand of numeric type [0 through 08, 10 and 11] and of length zero, shall be interpreted as the value zero.

A destination BDP operand of length zero shall transform the associated instruction into a no-op. However, exception sensing for the source field shall occur normally, including the testing for a Decimal Significance Loss condition, provided the source field does not also have a length of zero.

Note: The representations for zero, zones and signs shall be normally determined by interpreting the T field from the Data Descriptor associated with the destination field.

Division by zero shall not be allowed to the extent that the destination field in central memory shall not be changed and a Divide Fault condition shall be detected.

Minus zero shall be considered equivalent to plus zero by all the instructions in this subgroup, with respect to decimal numeric data.

NCR/CDC PRIVATE

For the instructions in this subgroup, (namely the Decimal instructions), as well as all the instructions contained in the BDP Instruction group, the term "D(Aj)" shall be used to denote "the contents of the source data fields," addressed by means of the components associated with the BDP instruction's j field designators. Similarly, the term "D(Ak)" shall be used to denote "the contents of the destination data field," addressed by means of the components associated with the BDP instruction's k field designators.

2.3.3.1 Arithmetic

a. Decimal Sum, D(Ak) replaced by D(Ak) plus D(Aj)
   074 jkØ/2
b. Decimal Difference, D(Ak) replaced by D(Ak) minus D(Aj)
   075 jkØ/2
c. Decimal Product, D(Ak) replaced by D(Ak) times D(Aj)
   076 jkØ/2
d. Decimal Quotient, D(Ak) replaced by D(Ak) divided by D(Aj)
   077 jkØ/2

Operation: These instructions shall arithmetically modify the initial contents of the destination field in central memory, (treated as an augend, minuend, multiplicand or dividend as determined by the operation code) by the contents of the source field in central memory (treated as an addend, subtrahend, multiplier or divisor as determined by the operation code) and shall transfer the decimal result consisting of a sum, difference, product or quotient, as determined by the operation code, to the destination field in central memory.

Types: All Packed decimal types and all Unpacked decimal types, except for the Leading Sign formats, shall be freely allowed for decimal arithmetic; i.e., types 0, 1, 2, 3, 4, 5, 6, shall be compatible for these instructions.

Unpacked Decimal Leading Sign (both conventions) shall not be supported in the decimal arithmetic. A Numeric Move instruction must be generated to format the operands of those types prior to their use in arithmetic operations.

Lengths: The maximum allowable lengths for the source and destination fields shall be determined according to their respective decimal data types as defined in subparagraph 2.3.2.3, item c, of this specification.

NCR/CDC PRIVATE

Notes: Decimal operands shall be treated as integer values. Division by zero shall be illegal to the extent that such operations shall be treated as Divide Faults in the manner described in paragraph 2.3.3 of this specification.

The results from these instructions shall be algebraically signed unless they are equal to zero in their entirety, in which case their signs shall be made positive.

These instructions shall generate a result value in accordance with the type T of the destination field and the preferred sign convention for that given type.

2.3.3.2 Shift

The following instructions shall move the data initially contained in the source field to the destination field. Unless the 32-bit halfword contained in Register XØ Right consists entirely of zeros in its rightmost 8-bit positions, these instructions shall also shift the data from the source field as it is moved to the destination field.

The 32-bit halfword contained in Register XØ Right shall be interpreted as a signed binary integer providing a signed shift count. When this 32-bit halfword is positive, the direction of the shift shall be left with the number of decimal digit positions to be shifted determined by the value of the rightmost 8 bits, in bit positions 56 through 63, of Register XØ Right. When this 32-bit halfword is negative, the direction of the shift shall be right with the number of decimal digit positions to be shifted determined by the value of the two's complement of the rightmost 8 bits, in bit positions 56 through 63, of Register XØ Right. (Once the direction of the shift has been determined, bit positions 32 through 55 of the shift count shall be ignored) Thus, positive shift counts shall provide the means for multiplying the source data field by powers of 10, and negative shift counts shall provide the means for dividing the source data field by powers of ten, as the source data is moved to the destination field.

When non-zero digits are shifted left end-off, or truncated on the left, a Decimal Significance Loss condition shall be detected.

Shifting shall be accomplished end-off with zero fill on the appropriate end(s) as required to accommodate the length and type of the receiving field. (For example, when the destination field is longer than the source field, and the difference in field lengths is greater than the left shift count, such a scale instruction shall provide zero fill, to the extent required, on both the right and left ends of the destination field result.)

NCR/CDC PRIVATE

Types: All packed decimal types and all unpacked decimal types, except for the Leading sign formats, shall be freely allowed for decimal scale instructions; i.e., types 0, 1, 2, 3, 4, 5, and 6 shall be freely compatible for these instructions.

Lengths: The maximum allowable lengths for the source and destination fields shall be determined according to their respective decimal data types as defined in subparagraph 2.3.2.3, item c, of this specification.

a. Decimal Scale, D{Ak} replaced by D{Aj}, scaled per {XD} Right.
   078 jkø/2
b. Decimal Scale Rounded, D{Ak} replaced by rounded D{Aj}, scaled per {XD} Right.
   079 jkø/2

Operation: These instructions shall move and scale, according to the contents of Register XD Right, the decimal data field initially contained in the source field to the destination field. They shall transfer the sign of the source field to the destination field without change, {unless the results consist entirely of zeros in which case the sign of the destination field shall be made positive or unless the result would otherwise contain a non-preferred sign in which case the sign of the destination field shall contain the preferred sign}.

Scale Rounded: When specified by means of the operation code, rounding shall be performed for negatively signed scale factors by adding five to the last digit shifted end-off and propagating carries, if any, through the decimal result transferred to the destination field. Thus the absolute value shall be rounded upwards.

NCR/CDC PRIVATE

2.3.3.3 Move DECIMAL AND BINARY

Numeric Move, D{Ak} replaced by D{Aj} after formatting
092 jkø/2

This instruction shall format the number obtained from the source field and shall transfer the result to the destination field.
The source field shall be validated according to the T field from its associated descriptor; the source field shall be reformatted according to the T field from the data descriptor associated with the destination field and the result shall be transferred to the destination field.

The format of the different data types allowed in this instruction are described in subparagraph 2.3.2.5 of this document. The conversion and format operation shall be performed on any combination of fields of type 0 through 11. In this instruction, an operand of type 9 shall be handled exactly as an operand of type 4, including length restriction, zero-fill, and data validation conventions as well as right to left processing.
If the source has a decimal data type and the destination a binary data type, a conversion from decimal to binary shall be performed. In this case, the maximum length for the source shall be determined by the decimal data type: 19 bytes for Types 0 through 3, and 38 bytes for Types 4 thru 8; the maximum field length for the destination shall be 8 bytes.
If the destination field is not long enough to accommodate the entire binary number, truncation of the leftmost bytes shall occur. If the destination field is longer than the result of the conversion, the sign bit shall be extended on the left.

If the source has a binary data type and the destination a decimal data type, a conversion from binary to decimal shall be performed. The length restrictions on the operands are the same as in the previous case. If the destination field is too short to accommodate the converted number, leading digits shall be truncated according to the destination's data type. If the receiving field is longer than the converted number, leading zeros shall be supplied in accordance with the decimal data type: ASCII character zero {hex{30}} or digit zero {hex{0}}.

When truncation of binary data results in loss of significance or an improper sign, an Arithmetic Overflow condition shall be recorded. When truncation of decimal data results in loss of significance, a Decimal Loss of Significance condition shall be recorded.

NCR/CDC PRIVATE

When both operands are decimal, their maximum allowable lengths shall be determined according to their respective decimal data types as defined in subparagraph 2.3.2.3, item c, of this specification.

The destination shall be filled from right to left. Unequal field lengths shall result either in truncation of the leading digits or in insertion of leading zeros according to the destination data type: ASCII character zero {hex{30}} or digit zero {hex{0}}.

2.3.3.4 Comparison

Decimal Compare D{Aj} to D{Ak}, result to X1 Right.
083 jk@/2

This instruction shall algebraically compare the decimal contents of the source field to the decimal contents of the destination field and shall transfer a 32-bit halfword to Register X1 Right according to the results of the comparison.

When the contents of the source and destination fields are equal, the entire 32-bit positions of Register X1 Right shall be cleared.

When the contents of the source field are greater than the contents of the destination field, Register X1 Right shall be cleared in the leftmost 31 bit positions, 32 through 62, and shall be set in the rightmost bit position, 63.

When the contents of the source field are less than the contents of the destination field, Register X1 Right shall be set in all 32 bit positions.

Types: All Packed decimal types and all Unpacked decimal data types except for the Leading Sign formats, shall be freely allowed in comparisons; i.e., types 0, 1, 2, 3, 4, 5, 6 shall be compatible for this instruction.

Lengths: Lengths shall be confined to the same maximum values as for a Decimal Difference instruction. Unequal field lengths shall be accommodated by providing zero fill in the leftmost positions, as required, for the field having the shorter length. The maximum number of bytes occupied by each operand is a function of its data type and is specified in subparagraph 2.3.2.3, item c, of this specification.

Note: As previously stated in 2.3.3, minus zero shall be interpreted as being equal to plus zero.

NCR/CDC PRIVATE

2.3.4 Byte

The instructions in this subgroup shall provide the means for comparing, scanning, translating, moving, and editing byte fields in central memory to the extent defined by the following instruction descriptions.

These instructions shall utilize spaces for extending Alphanumeric {Type 9} fields, with the space being represented by hex {20}.

A source byte operand of length zero shall be functionally interpreted as a string of space characters {ASCII character: hex {20}} for all the instructions in this subgroup.

A destination byte operand of length zero shall transform the associated instruction into a no-op. However, exception sensing for non-zero length fields shall occur normally, despite the destination field length of zero.

Decimal Significance Loss shall not be detected for the instructions in this subgroup.

2.3.4.1 Comparison

a. Byte Compare, D{Aj} to D{Ak}, result to X1 Right, index to X0 Right
084 jk@/2
b. Byte Compare Collated D{Aj} to D{Ak}, both translated per {A7}, result to X1 Right, index to X0 Right
085 jk@/2

These instructions shall compare the bytes contained in the source field to the bytes contained in the destination field and shall transfer the results of that comparison to Register X1 Right.

The comparison shall proceed from left to right. When the field lengths are unequal, trailing space characters shall be used for the field having the shorter length. The maximum length for each operand shall be 256 bytes.

These instructions shall ignore the Type field. Each byte from the source and destination field shall be treated as an 8-bit quantity having an absolute value with respect to the comparison operation.

NCR/CDC PRIVATE

The comparison shall continue until the longer field has been exhausted or until an "inequality" is detected between corresponding bytes from the source and destination fields according to the following definitions. For the Compare instruction, inequality between the bytes obtained directly from the source and destination fields shall result in the completion of the comparison. For the Collated Compare instruction inequality of the bytes obtained directly from the source and destination fields shall result in the translation of both bytes, by means of a translation table, and inequality of the translated bytes shall result in the completion of the comparison. When the translated bytes are equal, and the longer field has not been exhausted, comparison between the corresponding bytes obtained directly from the source and destination fields shall be resumed.

When every byte associated with the source field is equal to every corresponding byte associated with the destination field, (including the trailing space characters if any), the entire 32-bit positions of Register X1 Right shall be cleared. When the first inequality between bytes occurs as a result of a byte associated with the source field having a greater value than the corresponding byte associated with the destination field, Register X1 Right shall be cleared in the leftmost 31-bit positions, 32 through 62, and shall be set in the rightmost bit position, 63. When the first inequality between bytes occurs as a result of a byte associated with the source field having a value less than the corresponding byte associated with the destination field, Register X1 Right shall be set in all 32-bit positions. In addition, the sequence number of the byte which caused the 1st inequality will be placed in Register X0 Right.

Translation Table: The translation table used for each occurrence of direct inequality during Collated Compare instructions, shall be addressed by means of Address Register A7.

Each byte shall be translated by using its value as a positive offset to be added to the beginning (leftmost) address of the Translation Table, as contained in Register A7, for the purpose of addressing the translated byte to be read from central memory.

NCR/CDC PRIVATE

2.3.4.2 Byte Scan Count sequence # in field

Byte Scan While Non-Member, D[Ak] for presence bit in D[Aj], index to X0 Right, character to X1 Right.
086 jkQ/2

Operation: The operation shall proceed from left to right on the destination field addressed by D[Ak]. One character at a time shall be taken from this character string and used as a bit address into the string addressed by D[Aj]. The scan shall terminate if the bit thus addressed is ON or if the destination field has been exhausted; otherwise the next character in D[Ak] is considered.

Source Field: The type and length fields in D[Aj] shall be ignored. The operand addressed by D[Aj] shall be interpreted as a bit string consisting of 256 bits (32 bytes).

Destination Field: The type field in D[Ak] shall be ignored. The operand addressed by D[Ak] shall be interpreted as a byte string, and restricted to no more than 256 characters.

The binary value of the sequence number in the string, of the byte which caused the scan to terminate shall be placed right justified into X0 Right.

The binary value of the character itself which caused the scan to terminate shall be placed right justified into X1 Right.

If the scan stops by exhaustion of the characters in the byte string, X0 Right shall contain the length of the original byte string and X1 Right shall be set equal to -1.

Note: The function Byte Scan While Member can be performed by means of the Byte Scan While Non-Member if the bit string specifying the characters not allowed in the byte string has been previously logically negated.

Add the value obtained = D[Ak] to determine Bit address of D(Aj) and check for ones.

NCR/CDC PRIVATE

2.3.4.3 Translate

Byte Translate, D{Ak} replaced by D{Aj}, translated per A7
088 jk@/2

This instruction shall translate each byte contained in the source field, according to the translation table in central memory and shall transfer the results of the byte by byte translation to the destination field.

The translation table shall be addressed in a manner identical to that previously described for the Collated Compare instruction in subparagraph 2.3.4.1 of this specification.

The Type fields in the Data Descriptors associated with the source field and the destination field shall be ignored. Both operands shall be restricted to no more than 256 bytes.

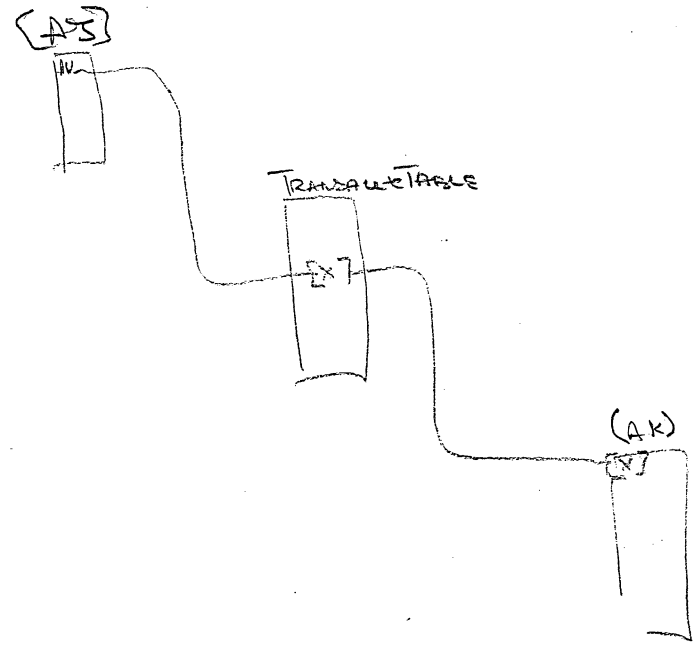
The translation operation shall occur from left to right with each source byte used as a positive offset to be added to the beginning {leftmost byte} address of the translation table for the purpose of permitting each byte's translation. Translated bytes, thus obtained from the translation table, shall be transferred to the destination field. The translation operation shall terminate after the destination field length has been exhausted. When the source field length is greater than the destination field length, rightmost bytes from the source field shall be truncated, to the extent required, with respect to the translation operation. When the source field length is less than the destination field length, translated space characters shall be used to fill the rightmost byte positions of the destination field to the extent required.

2.3.4.4 Move

This instruction shall provide the means for moving the bytes contained in the source field to the destination field. The type fields of the source and destination data descriptors shall be ignored and any combinations of the values 0 through 11 shall be allowed.

Move Bytes, D{Ak} replaced by D{Aj}.
089 jk@/2

This instruction shall move the bytes contained in the source field to the destination field. The operation shall be performed from left to right with unequal field lengths accommodated by the truncation of trailing characters from the source field or the insertion of trailing spaces into the destination field.



2.3.4.5 Edit

Edit, D{Ak} replaced by D{Aj} edited per M{A7}.
091 jk0/2

This instruction shall edit the digits or characters contained in the source field according to an edit mask in central memory and shall transfer the result to the destination field. The edit mask shall be addressed by the PVA contained in A7. The edit mask shall consist of a one byte length indication followed by a string of micro-operations.

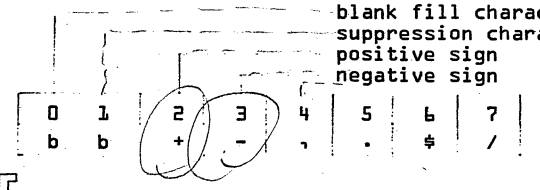
The edit instruction shall terminate under control of the mask. If the destination field is filled before the end of the mask is encountered, then the writing of further characters beyond the destination field shall be disabled but the Edit shall proceed until the end of the mask is encountered; an Invalid BDP Data condition shall be recorded. If the Edit terminates before the destination field is filled, then any further characters in the destination field shall remain undisturbed and no exception conditions shall be detected for these circumstances.

Type: The Data Descriptor type fields shall be confined to the following:

Source	Destination
0,1,2,3,4,5,6,7,8,9	9

Special Conventions: The edit operation shall utilize the tables and toggles listed below.

a. Special Characters Table {SCT}: The SCT is an eight byte table that shall be initialized by the machine at the start of each edit operation to contain the following.



NCR / CDC PRIVATE

Entries in the SCT shall be readable/writable under control of certain micro-operations comprising the mask.

b. The Symbol {SM}: The symbol is a string of 0 to 15 characters that shall be created under control of the edit mask and inserted into the destination field under control of the edit mask. Once the SM has been inserted into the destination field, it must be recreated before it can be inserted again. At the start of an edit operation, the SM shall have a zero length.

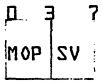
The SM shall be utilized for the floating sign and floating currency editing features. It shall also be utilized for sign sensitive and significance sensitive character string insertion.

c. End Suppression Toggle {ES}: This toggle controls zero suppression. At start of edit, the ES shall be initialized FALSE. The ES shall be set TRUE when zero suppression ends.

d. Negative Sign Toggle {SN}: This toggle signifies the sign of the source field. At start of edit, the SN shall be initialized FALSE for an alphanumeric source field or a positive numeric source field. It shall be initialized TRUE for a negative numeric source field.

e. Zero Field Toggle {ZF}: This toggle signifies whether the source field is zero or non-zero. It shall be initialized FALSE for an alphanumeric source field and TRUE for a numeric source field.

Edit Micro-Operations: The mask shall be interpreted as a string of one byte micro-instructions with the following format.



The MOP is a micro-operator. It specifies an editing function. The SV is a specification value. It's meaning varies according to the specific MOP which it follows.

NCR / CDC PRIVATE

Edit control shall proceed from left to right on the mask, one character {or micro-operation} at a time. After interpretation of the micro-operation, action shall be taken on the source and destination field characters {or source digits} which shall also be operated from left to right.

Indexing through the source field shall be by bytes unless its data-type is packed numeric when indexing shall be by half-bytes. Indexing through the destination field shall be by bytes.

What happens when you are to move the sign.

Notation for MOP descriptions.

i	Index for source field, initialized to 0.
j	Index for destination field, initialized to 0.
k	Index for mask, initialized to 0.
SC{i}	The source character addressed by base of source field indexed by i.
SD{i}	The source digit addressed by base of source field indexed by i.
DC{j}	The destination byte addressed by base of destination field indexed by j.
MC{k}	The mask byte addressed by base of mask field indexed by k.
ES	End suppression toggle.
ZF	Zero field toggle.
SN	Sign toggle.
SCT	Special character table
SCT{n}	Nth entry in the SCT {value must be 0-7}.
SV	Specification value.
SM	The symbol.
LSM	Length of the Symbol in bytes, initialized to zero.
R	A loop counting mechanism {associated with SV}.
C	A loop counting mechanism {associated with LSM}.

Note: The one byte length indication contained in the leftmost byte position of the Edit Mask shall include itself in specifying the length of the Edit Mask. {Thus, a maximum of 254 micro-operations may be specified by this byte}.

Although not included in each description, prior to the execution of each micro-operation the edit mask index shall be incremented by one.

The following descriptions of the micro-operations are not intended to rigidly dictate the implementation technique so long as the function of each individual MOP is accomplished.

The logic is specified concisely using a SWL-like notation.

a. Move Source Digits {MOP=0}
SV may be a value 0-15

```
ES := TRUE;
FOR R := 1 to SV DO
   DC{j} := NUMERIC {SD{i}};
   i := i+1; j := j+1; FOREND;
```

SOURCE FIELD DATA TYPE 9 SHALL BE TREATED AS TYPE 4 ?

b. Move Source Characters {MOP=1}
SV may be a value 0-15

```
ES := TRUE;
FOR R := 1 to SV DO
   DC{j} := SC{i};
   i := i+1; j := j+1; FOREND;
```

{Ignore type}

See item r for special case.

SOURCE FIELD DATA TYPE SHALL BE IGNORED AND TREATED AS TYPE 9.

If Source 0-9 SET INVALID

c. Skip Source Positions {MOP=2}
SV may be a value 0-15

```
i := i+SV;
```

THE SIZE OF EACH SOURCE POSITION SHALL BE DETERMINED BY THE SOURCE FIELD DATA TYPE.

d. Skip Destination Positions {MOP=3}
SV may be a value 0-15

```
ES := FALSE;
j := j+SV;
```

e. Move Mask Characters {MOP=4}
SV may be a value 0-15

```
FOR R := 1 TO SV DO
   DC{j} := MC {k+1};
   j := j+1; k := k+1; FOREND;
```

f. Select a Sign as the Symbol {MOP=5}
SV may be a value 0-F {but is normally 1 or 2}.

SV shall be evaluated by ignoring the high order bit.

```
LSM := 1;
IF SN THEN   SM[0] := SCT[3];        "NEGVE.='-'"
     ELSE    SM[0] := SCT[SV] IFEND; "POSVE.='+' or 'b'"
```

g. Select Mask Characters as the Symbol {MOP=6}
SV may be a value 0-15

```
LSM := SV
FOR C := 1 to LSM DO

   SM[C-1] := MC[k+1];
   k := k+1; FOREND;
```

h. Move Source Digits or Suppress with Floating Symbol {MOP=7}
SV may be a value 0-15

```
FOR R := 1 TO SV DO

   IF NOT ES THEN

      IF SD [i] = 0 THEN

         DC[j] := SCT[1];                "SUPPRESS"
         i := i+1; j := j+1;

      ELSE "IF SD [i] ≠ 0"               "1ST SIGNIF. DIGIT"

         FOR C := 1 TO LSM DO
         DC[j] := SM[C-1];               "INSERT SYMBOL"
         j := j+1; LSM := LSM-1;
         FOREND;
         DC[j] := NUMERIC [SD[i]];
         i := i+1; j := j+1;
         ES := TRUE; IFEND;

   ELSE "IF ES"                          "INSERT DIGITS"

      DC[j] := NUMERIC [SD[i]];
      i := i+1; j := j+1; IFEND;

FOREND;
```

NCR/CDC PRIVATE

i. End Float {MOP=8}
SV is not used

```
IF NOT ES THEN

   ES := TRUE;
   FOR C := 1 TO LSM DO
   DC[j] := SM[C-1];
   j := j+1; LSM := LSM-1;
   FOREND; IFEND;
```

j. Insert Symbol or SCT Character, Unconditional {MOP=9}
SV may be a value 0-15.

```
IF SV >7 THEN

   FOR C := 1 TO LSM DO
   DC[j] := SM[C-1];
   j := j+1; LSM := LSM-1;
   FOREND;

ELSE "IF SV <8 THEN"

   DC[j] := SCT[SV];
      j := j+1; IFEND;
```

k. Insert Symbol or SCT Character if Posve., Else blanks {MOP=10}
SV may be a value 0-15

```
IF SV >7 THEN

   FOR C := 1 TO LSM DO
   IF NOT SN THEN DC[j] := SM[C-1];
   ELSE           DC[j] := SCT[0]; IFEND;
   j := j+1; LSM := LSM-1;
   FOREND;

ELSE "IF SV <8 THEN"

   IF NOT SN DC[j] := SCT[SV];
   ELSE      DC[j] := SCT[0]; IFEND;
             j := j+1
IFEND;
```

NCR/CDC PRIVATE

l. Insert Symbol or SCT Character if Negve., Else blanks {M0P=11}
SV may be a value 0-15

This M0P is similar to M0P 10. Substitute IF SN for IF NOT SN in that algorithm.

m. Insert Symbol or SCT Character, Unless Suppression {M0P=12}
SV may be a value 0-15

```
IF SV >7 THEN

   FOR C := 1 to LSM DO
   IF ES THEN DC{j} := SM{C-1};
   ELSE  DC{j} := SCT{1}; IFEND;                    "SUPPRESS"
   j := j+1; LSM := LSM-1;
   FOREND;

ELSE  "IF SV < 8 THEN"

   IF ES DC{j} := SCT {SV};
   ELSE  DC{j} := SCT{1}; IFEND;                    "SUPPRESS"
         j  := j + 1
IFEND;
```

n. Write SCT Entry {M0P=13}
SV may be a value 0- F

SV shall be evaluated by ignoring the high order bit.

```
SCT{SV} := MC{k+1};
     k  := k+1;
```

o. Spread Suppression Character {M0P=14}
SV may be a value 0-15

```
FOR R := 1 TO SV DO

   DC{j} := SCT{1}; j := j+1; FOREND;
```


p. Reset and Suppress, On Zero Field {M0P=15}
SV may be a value 0-15

```
IF ZF THEN

   j := 0                          "RESET TO START OF DEST. FIELD"
   FOR R := 1 TO SV DO

      DC{j} := SCT{1}; j := j+1; FOREND;

ELSE RETURN;                       "IE. TERMINATE EDIT
                                        INSTRUCTION"
IFEND;
```

q. Function NUMERIC

This function shall be used by micro-operations 0 and 7 to move a source digit into the destination field.

Each source digit shall be checked, invalid decimal digits shall cause an Invalid BDP Data condition to be detected and, if enabled, a program interruption shall occur upon the completion of the Edit instruction.
When the source field is packed numeric, appropriate ASCII zone bits shall be supplied for the destination character.

A non-zero digit shall cause the ZF toggle to be set FALSE.

r. Special Case, M0P = 1

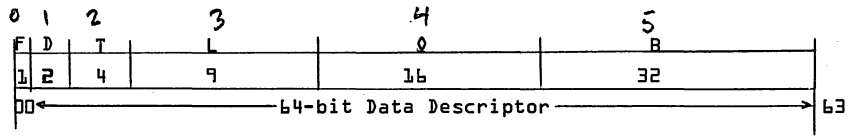
When SV is not equal to zero, and i would not reference a byte boundary, it shall be increased by one prior to beginning the operation.

2.3.5 Descriptor

The instructions in this subgroup shall provide the means for moving or algebraically modifying Data Descriptors, or Data Descriptor fields, to the extent defined in the following subparagraphs.

The fields within a 64-bit Data Descriptor shall be described singly and in limited combinations by means of a 3-bit field designator, having values from 0 through 7, as follows:

Field Designator	64-bit Data Descriptor fields{s}
0	F
1	D
2	T
3	L
4	Q
5	B
6	F,D,T,L and Q
7	F,D,T,L,Q and B



For these instructions, the j1 and/or the k1 fields shall be interpreted as field designator with respect to the associated Data Descriptor Operations.

Note: For the instructions in this subgroup, "F{j1}" or "F{k1}" represent the descriptor field specified by the value of the field designator in j1 or k1, and "D{j0,Q0}" or "D{k0,Q1}" represent the descriptor in the DDT specified by j0 or k0 and whose offset within the DDT is given by Q0 or Q1.

NCR/CDC PRIVATE

2.3.5.1 Move Table Entry

Descriptor Table Entry Move, F{k1} of D{k0,Q1} replaced by F{j1} of D{j0,Q0}
094 jkQ/2

This instruction shall move the Data Descriptor Table Entry associated with the source field to the Data Descriptor Table entry associated with the destination field according to the 3-bit designators j1 and k1 from the instruction.

The interpretation of the j1 and k1 designators shall be performed in the manner described in paragraph 2.3.5 of this specification with respect to designating the source and destination fields within the corresponding Data Descriptors.

These field designator values shall be confined to the following combinations:

j1 {Source}	k1 {Destination}
0 or 5	0
1 or 5	1
2 or 5	2
3 or 5	3
4 or 5	4
5 or 6	5
5 or 6	6
7	7

Unequal field lengths shall be accommodated by performing truncation in the leftmost bit positions, to the extent required.

2.3.5.2 Increment Descriptor fields from Table Entry

Descriptor Table Entry Increment, F{k1} of D{k0,Q1} replaced by F{k1} of D{k0,Q1} plus F{j1} of D{j0,Q0}

095 jkQ/2

This instruction shall add selected fields of the Data Descriptor associated with the source field to selected fields of the Data Descriptor associated with the destination field and shall transfer the result to the Data Descriptor Table entry associated with the destination field.

The interpretation of the 3-bit designators j1 and k1 from the instruction, shall be performed in the manner described in paragraph 2.3.5 of this specification, with respect to designating the participating fields within the associated Data Descriptors.

The field designator values shall be confined to the following combinations:

j1 {Source}	k1 {Destination}
0 or 5	0
1 or 5	1
2 or 5	2
3 or 5	3
4 or 5	4
5 or 6	5
5 or 6	6
7	7

Unequal field lengths shall be accomodated by performing truncation in the leftmost bit positions, to the extent required.

2.3.5.3 Move fields between X-Register and Descriptor

Descriptor Field Insert from X-Register, F{k1} of D{k0,Q1} replaced by {Xj}
146 jkQ/2

Operation: This instruction shall move the contents of register Xj into the field specified by k1, within the Data Descriptor specified by D{k0,Q1}.

Source: For the purposes of this instruction, the entire j field {i.e., j0 concatenated with j1} shall be interpreted as the X-Register designator. The source data shall be assumed right justified in the X-Register.

Destination: The Data Descriptor normally associated with the destination field shall serve as the destination field itself for the instruction. The 3-bit k1 designator from this instruction shall be interpreted in the manner described in paragraph 2.3.5 of this specification with respect to designating the field within the destination Data Descriptor to which the source field data shall be transferred. Unequal field lengths shall be accommodated by performing truncation in the leftmost bit positions of the source data field to the extent required.

Descriptor Field Extract to X-Register, Xk replaced by F{j1} of D{j0,Q0}
147 jkQ/2

Operation: This instruction shall move the contents of the field specified by F{j1} within the Data Descriptor specified by D{j0,Q0} into register Xk.

Source: The Data Descriptor normally associated with the source field shall serve as the source field itself for this instruction. The 3-bit j1 designator determines which fields are to be extracted from the Data Descriptor {see paragraph 2.3.5.}. The contents of the specified fields shall be treated as binary data to be moved right justified to the destination X-Register.

NCR / CDC PRIVATE

Destination: For the purposes of this instruction, the entire k field {i.e., k0 concatenated with k1} shall be interpreted as the X-Register designator. The X-Register, X0- XF, so specified shall receive the binary data right justified. When extracting individual fields to Xk, signed fields {i.e., 0 and B fields} shall be sign extended to the left. When extracting unsigned fields or combinations of fields to Xk, the unused portion of the X-Register shall be zero filled to the left.

2.3.5.4 Increment and Decrement Descriptor fields by X-Register

Descriptor Field Increment by X-Register, F{k1} of D{k0,Q1} incremented by {Xj} Right
148 jkQ/2

Descriptor Field Decrement by X-Register, F{k1} of D{k0,Q1} decremented by {Xj} Right
149 jkQ/2

Operation: According to the Op code, these instructions shall add/subtract in signed binary arithmetic, the contents of the source register Xj Right to the field specified by F{k1} within the Data Descriptor specified by D{k0,Q1}.

Source: For the purposes of this instruction, the entire j field {i.e., j0 concatenated with j1} shall be interpreted as the X-Register designator. The source data shall be assumed signed binary right justified in Xj Right.

Destination: The Data Descriptor normally associated with the destination field shall serve as the destination field itself for this instruction. The 3-bit k1 field shall determine which field is to be incremented/decremented {see paragraph 2.3.5}. The only legal values for k1 are 3, 4 or 5. Thus, when k1 is not equal to 3, 4 or 5, an Instruction Specification error shall be detected, the execution of the instruction shall be inhibited and a program interruption shall occur.

The result shall be truncated on the left to match the length of the destination field.

NCR / CDC PRIVATE

2.3.5.5 Calculate Subscript

Calculate Subscript and Move per D{Aj} times {X0} Right, result moved to D{k0,Q1} and to X1 Right.
096 jkQ/2

This instruction shall obtain a 32-bit, two's complement binary integer from the source field in central memory, either directly for binary source field data, or by converting decimal source field data to its binary equivalent. The signed, 32-bit halfword thus obtained, shall be multiplied by the positive 32-bit halfword contained in Register X0 Right. The algebraically signed result shall consist of a 32-bit binary integer which shall be transferred to both the Data Descriptor Table entry associated with the destination field and Register X1 Right.

Source Field: The Data Descriptor associated with the source field shall be confined to Type field values of 0 through 6, 10 and 11 with the maximum Length field value determined by the Source field data Type as defined in subparagraph 2.3.2.3, item c, of this specification.

Exceptions: When the 32-bit halfword contained in Register X0 Right is negative an Instruction Specification error shall be detected. Likewise, when either the conversion or multiplication phases of instruction execution result in a signed, 32-bit binary integer outside of the range from -2^{31} to $2^{31}-1$ an Instruction Specification error shall be detected. In this context, the detection of an Instruction Specification error shall result in inhibiting the execution of this instruction and the corresponding program interruption shall occur.

Destination Field: The Data Descriptor Table entry normally associated with the destination field shall serve as the destination field itself for this instruction. The positive 32-bit binary integer resulting from the multiplication phase of instruction execution, shall be transferred to central memory at F{b} of D{k0,Q1} and Register X1 Right.
The 3-bit k1 designator from this instruction and the initial value of the Data Descriptor Table entry associated with the destination field shall not be translated by this instruction.

Unless the PVA of the Data Descriptor associated with the destination field is equal to 0, modulo 4, an Address Specification Error shall be detected, the execution of this instruction shall be inhibited, and a program interruption shall occur.

NCR/CDC PRIVATE

2.3.6 Logical: Memory to Memory

See paragraph 2.2.10 of this specification.

NCR/CDC PRIVATE

2.3.7 Immediate Data

2.3.7.1 Move Immediate Data to D{Ak}.

154 jkQ/2

Operation: This command shall move the explicit value contained in the Q0 field to the destination field after an eventual format conversion to match the destination type.

Source: The Q0 field shall contain the literal to be moved. The least significant 2 bits of the j field shall be used as an encoding of the operation to be performed:

a. if = 00, the unsigned {considered positive} numeric value {Type 10} contained in Q0 shall be moved right justified to the receiving field which must be of type 10 or 11. If necessary, the destination field is filled with zeros on the left.

b. if = 01, the decimal numeric {Type 4} contained in Q0 shall be moved right justified, to the receiving field after possible reformatting to match the data type of the destination. If the format requires a sign, a positive sign shall be supplied. The destination shall be restricted to one of the decimal data types 0, 1, 2, 3, 4, 5, or 6. This move shall be executed according to the rules of the numeric move for truncation, padding and validation.

c. if = 10, the ASCII character contained in Q0 is repeated left to right in the receiving field. The destination data type shall be ignored.

d. if = 11, the ASCII character contained in Q0 is moved left justified into the receiving field, and the rest of that field is space filled. The destination data type shall be ignored.

2.3.7.2 Compare Immediate Data to D{Ak}
155 jkQ/2

This command shall, depending on the value of the j field, compare the explicit value contained in the Q0 field to D{Ak} after a possible reformatting to match the data type and shall transfer a 32-bit halfword to Register X1 right according to the result of the comparison.

When the contents of the source and destination fields are equal, the entire 32-bit positions of Register X1 Right shall be cleared.

NCR/CDC PRIVATE

When the contents of the source field are greater than the contents of the destination field, Register X1 Right shall be cleared in the leftmost 31-bit positions, 32 through 62, and shall be set in the rightmost bit position, 63.

When the contents of the source field are less than the contents of the destination field, Register X1 Right shall be set in all 32-bit positions, 32 through 63.

The interpretations of the source and destination fields are analogous to those described under the Move Immediate Data instruction.

2.3.7.3 Add Immediate Data to D{Ak}
156 jkQ/2

Operation: This command shall add the explicit integer value contained in the Q0 field to D{Ak} after a possible conversion to match the destination data type.

Source: The Q0 field is used to store the integer value of the addend.
The j field is used as an encoding of the type of the data contained in Q0. The least significant bit of the j field is decoded as follows:

a. if = 0, Q0 contains an unsigned {considered positive} binary integer value; Q0 = Data Type 10

b. if = 1, Q0 contains one ASCII character representing a decimal digit; if invalid decimal data is encountered in Q0, an Invalid BDP Data condition shall be detected and upon completion of the instruction's execution, the corresponding program interruption shall occur, if enabled; Q0 = Data Type 4.

If the source corresponds to case a. above, the destination shall be confined to types 10 and 11.

If the source corresponds to case b. above, the destination shall be confined to types 0, 1, 2, 3, 4, 5, or 6.

If unauthorized data types are specified, an Instruction Specification error shall be detected, the instruction's execution shall be inhibited, and the corresponding program interruption shall occur. See 2.8.1.4.

NCR/CDC PRIVATE

2.3.8 Register Loads

2.3.8.1 Load A Register

Load Ak with computed address per descriptor D{Aj}
157 jkØ/2

Operation: This instruction shall compute the PVA corresponding to D{Aj} and place it into Ak. For the purposes of this instruction, the entire k field {i.e., kØ concatenated with k1} is interpreted as the A Register designator.

2.3.8.2 Load, Store X Register

a. Load Xk from {D{Aj}}
158 jkØ/2

Operation: This instruction shall load Register Xk right-justified with the value of the data item addressed by D{Aj}. If the data item is shorter than 8 bytes, Register Xk is zero filled according to the data type of D{Aj} or filled with sign bit extension if the data type of D{Aj} is 11.

Destination: For the purposes of this instruction, the entire k field {i.e., kØ concatenated with k1} shall be interpreted as the X Register designator.

Source: D{Aj} is the source field descriptor. Its length field is limited to a maximum value of 8 bytes.

Type 10.11 only

b. Store {Xj} into D{Ak}
159 jkØ/2

Operation: This instruction shall store the contents of Register Xj into the data item addressed by D{Ak}. If the destination field is shorter than 8 bytes, the most significant bytes or digits shall be truncated. If the destination field is longer than 8 bytes, it shall be zero filled on the left according to the data type, except if the type is 11 in which case the sign bit is extended.

Destination: D{Ak} is the destination field descriptor. Its length shall be limited to a maximum value of 8 bytes.

Source: For the purposes of this instruction, the entire j field {i.e., jØ concatenated with j1} shall be interpreted as the X-Register designator.

NCR/CDC PRIVATE

2.4 Floating Point Instructions

2.4.1 General Description

A floating point number shall consist of a signed exponent and a signed fraction. The signed exponent shall also be referred to as the characteristic and the signed fraction shall also be referred to as the coefficient.

The quantity expressed by a floating point number shall be of the form $\{f\}2^x$ where f represents the signed fraction and x represents the signed exponent of the base 2.

The exponent base of 2 shall be an implied constant for all floating point numbers and thus shall not explicitly appear in any floating point format.

2.4.1.1 Formats

Floating point data shall occupy one of two fixed length formats; 64-bit word {Single Precision} or 128-bit doubleword {Double Precision}.

In both the single and double precision formats, the leftmost bit position, ØØ, shall be occupied by the sign of the fraction. The fifteen bit positions immediately to the right of bit ØØ, Ø1 through 15, shall be occupied by the signed exponent.

The field immediately to the right of the signed exponent shall be occupied by the fraction which in single precision format shall consist of 48 bits and in double precision format shall consist of 96 bits, according to the following figures.

NCR/CDC PRIVATE

2.4 Floating Point Instructions

2.4.1 General Description

A floating point number shall consist of a signed exponent and a signed fraction. The signed exponent shall also be referred to as the characteristic and the signed fraction shall also be referred to as the coefficient.

The quantity expressed by a floating point number shall be of the form $\{f\}2^x$ where f represents the signed fraction and x represents the signed exponent of the base 2.

The exponent base of 2 shall be an implied constant for all floating point numbers and thus shall not explicitly appear in any floating point format.

2.4.1.1 Formats

Floating point data shall occupy one of two fixed length formats; 64-bit word {Single Precision} or 128-bit doubleword {Double Precision}.

In both the single and double precision formats, the leftmost bit position, 00, shall be occupied by the sign of the fraction. The fifteen bit positions immediately to the right of bit 00, 01 through 15, shall be occupied by the signed exponent.

The field immediately to the right of the signed exponent shall be occupied by the fraction which in single precision format shall consist of 48 bits and in double precision format shall consist of 96 bits, according to the following figures.

NCR/CDC PRIVATE

00	01	15	16	63
S	Signed Exponent		48-bit fraction	

Single Precision Floating Point Number

00	01	15	16	63
S	Signed Exponent		Leftmost 48 bits of the 96-bit fraction	

64	71	72	127
		Rightmost 48 bits of the 96-bit fraction	

Double Precision Floating Point Number

A double precision floating point number shall consist of two single precision floating point numbers located in consecutively numbered X Registers. The two single precision floating point numbers comprising a double precision floating point number shall be referred to as the leftmost and rightmost parts as contained in the Xn and Xn+1 Registers, respectively. The leftmost part may be any single precision floating point number and when it is normalized, (the leftmost bit of the fraction, in bit position 16, is equal to a one) the double precision floating point number shall be considered to be normalized. The sign of the fraction and the characteristic of the leftmost part shall constitute the sign of the fraction and the characteristic of the double precision number.

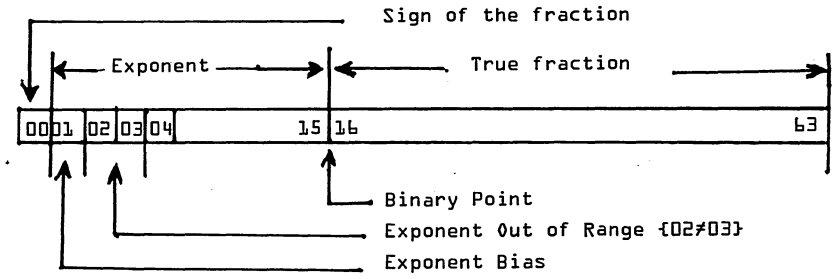
The fraction field of the leftmost part shall constitute the leftmost 48 bits of the 96-bit double precision fraction. The fraction field of the rightmost part shall constitute the rightmost 48 bits of the 96-bit double precision fraction. The sign of the fraction and the characteristic of the rightmost part shall not be utilized from any number constituting an input operand (argument) to a double precision floating point operation. Such operations shall assume that the sign of the fraction of the rightmost part is the same as the sign of the fraction of the leftmost part and that the characteristic of the rightmost part is

NCR/CDC PRIVATE

48 less than the characteristic of the leftmost part. However, the formation of a double precision floating point result shall include making the sign of the fraction of the rightmost part the same as that of the leftmost part and, except for certain cases involving non-standard forms of floating point results, shall also include making the characteristic of the rightmost part 48 less than the characteristic of the leftmost part. See 2.4.3.3.

2.4.1.2 Standard Numbers

The fraction field of a floating point number shall have its binary point immediately to the left of its leftmost bit position, 16. Both positive and negative quantities shall have a true fraction with the sign indicated solely by means of the sign bit. A number shall be positive or negative depending on whether the sign is a zero or a one, respectively.



The fraction shall be considered to be multiplied by the power of 2 expressed by the exponent which, in encoded form, occupies bit positions 01 through 15. The exponent field shall be used to represent both standard and non-standard floating point numbers. Standard floating point numbers shall have an actual exponent range from -4096 to 4095 inclusive, and shall be encoded into the exponent field by adding a bias equal to 2^{14}. The effect of biasing the exponent is demonstrated in Table 2.4-1 for standard floating point numbers in which the ascending order from smallest to largest encoded representations corresponds to the smallest to largest progression of multiplier values represented by the actual exponents in the range of -4096 to 4095 inclusive.

NCR/CDC PRIVATE

The ranges in magnitude, M, covered by standard, normalized floating point numbers in each of the two formats is as follows:

Single precision; $2^{-4097} \leq M \leq (1-2^{-48}) 2^{4095}$
(Approximately 14.4 decimal digits of precision)

Double precision; $2^{-4097} \leq M \leq (1-2^{-96}) 2^{4095}$
(Approximately 28.9 decimal digits of precision)

For both formats these ranges approximate to:

$(4.8) 10^{-1234} \leq M \leq (5.2) 10^{1232}$

2.4.1.3 Non-standard Numbers

The exponent field shall also be used to represent non-standard floating point numbers referred to as Zero, Infinity and Indefinite. Further, the exponent field shall be used to represent Out of Range values resulting from Exponent Overflow for actual exponents of 4096 to 12,287 inclusive, and Exponent Underflow for actual exponents of -4097 to -12,288 inclusive, provided such overflow or underflow conditions are enabled for a corresponding program interruption at the time such exceptions occur.

Table 2.4-1 illustrates hexadecimal exponent codes for corresponding non-standard as well as standard floating point numbers.

a.) Zero. Non-standard floating point numbers constituting input arguments to floating point operations shall be treated as if they consisted entirely of zeroes when (bit 01) and (bit 02 and/or 03) are equal to zeroes.

When non-standard floating point input arguments generate a zero result by definition and when the occurrence of exponent underflow in a floating point result is detected but not enabled with respect to the generation of the corresponding program interruption, such results shall consist entirely of zeroes. (Also, see 2.4.1.6, item c for add/subtract, special case).

b.) Infinity. Non-standard floating point numbers constituting input arguments to floating point operations shall be treated as infinite values when bit 01 is equal to one and bits 02 and 03 are not equal to each other.

	Hexadecimal Exponent Representation including coefficient sign	Actual exponent to the base 2	
	7XXX	—	Indefinite Input Arguments; Results=7000.0→0
	6FFF	2^{12287}	Infinite Input Arguments
	6000	2^{8192}	Exponent Overflow Results;
	5FFF	2^{8191}	Overflow Mask=0, Infinite Results=5000.0→0
	5000	2^{4096}	Overflow Mask=1, Out of Range Results as shown
Coefficient sign equal to 0 (positive numbers)	4FFF	2^{4095}	Standard Floating Point Arguments and Results
	4000	2^{0}	
	3FFF	2^{-1}	
	3000	2^{-4096}	
	2FFF	2^{-4097}	Zero Input Arguments
	2000	2^{-8192}	Exponent Underflow Results;
	1FFF	2^{-8193}	Underflow Mask=0, Zero Results=0000.0→0
	1000	$2^{-12,288}$	Underflow Mask=1, Out of Range Results as shown
	0XXX	0	Zero Input Arguments; Results=0000.0→0
	8XXX		
	9000	$2^{-12,288}$	Zero Input Arguments
	9FFF	2^{-8193}	Exponent Underflow Results;
	A000	2^{-8192}	Underflow Mask=0, Zero Results=0000.0→0
	AFFF	2^{-4097}	Underflow Mask=1, Out of Range Results as shown
Coefficient sign equal to 1 (negative numbers)	B000	2^{-4096}	Standard Floating Point Arguments and Results
	BFFF	2^{-1}	
	C000	2^{0}	
	CFFF	2^{4095}	
	D000	2^{4096}	Infinite Input Arguments
	DFFF	2^{8191}	Exponent Overflow Results;
	E000	2^{8192}	Overflow Mask=0, Infinite Results=D000.0→0
	EFFF	$2^{12,287}$	Overflow Mask=1, Out of Range Results as shown
	FXXX	—	Indefinite Input Arguments; Results=7000.0→0

Table 2.4-1: Floating Point Representation

When non-standard floating point input arguments generate an infinite result by definition and when the occurrence of exponent overflow in a floating point result is detected but not enabled with respect to the generation of the corresponding program interruption, such results shall consist entirely of zeroes except in bit positions 01 and 03 which shall be ones and bit position 00 which shall be a one or a zero as determined by the rules of algebra.

c.) Indefinite. Non-standard floating point numbers constituting input arguments to floating point operations shall be treated as indefinite values when bits 01 through 03 are all equal to ones.

When non-standard floating point input arguments generate an indefinite result by definition, such a result shall consist entirely of zeroes except for bits 01 through 03 which shall consist of all ones.

d.) Notes. When non-standard results are generated, as previously described by items a through c, the rightmost part shall be made identical to the leftmost part for all cases of double precision floating point results.

2.4.1.4 Exponent Arithmetic

When the exponent fields from input arguments are added, as for floating point multiplication, or subtracted, as for floating point division, the exponent arithmetic shall be performed algebraically in 2's complement mode. Moreover, such operations shall take place, conceptually, as if the bias were removed from each exponent field prior to performing the addition or subtraction and then restored following exponent arithmetic so as to correctly bias the exponent result.

Exponent Underflow and Overflow conditions shall be detected for all single precision, but only for the leftmost part of double precision floating point results. When the generation of the exponent of the rightmost part, by reducing the exponent of the leftmost part by 48, results in underflow for the rightmost part, this underflow shall not be detected and utilization of an Out of Range exponent shall permit the rightmost part of the double precision floating point number to correctly express its value.

NCR/CDC PRIVATE

2.4.1.5 Normalization

A normalized floating point number shall have a one in the leftmost bit position, 16, of the fraction field. If the leftmost bit of the fraction is a zero the number shall be considered unnormalized. The normalization process shall consist of left shifting the fraction until the leftmost bit position contains a one and correspondingly reducing the characteristic by the number of bit positions shifted. For double precision floating point numbers, the entire fraction shall participate in the normalization such that the rightmost part may or may not appear as a normalized single precision number as determined by the value of the fraction.

Normalization shall usually take place when intermediate results are changed to final results and shall be referred to as postnormalization. Normalization shall also take place prior to multiplication and division operations, if required, and shall be referred to as prenormalization.

For normalized operations, the input arguments shall not be required to be in normalized form. For both normalized and unnormalized operations, intermediate results in which overflow has occurred with respect to the fraction, shall be shifted right one bit position with the characteristic correspondingly increased by one, (to account for the right shift of the fraction).

Numbers with zero fractions cannot be normalized and such fractions shall remain equal to zero irrespective of normalization operations. Moreover, prenormalization of such fractions shall also leave their corresponding exponents unchanged.

When exponent arithmetic and/or prenormalization operations on standard floating numbers generate an intermediate exponent which is Out of Range, but postnormalization requirements generate an adjusted exponent which is no longer Out of Range, then neither Exponent Overflow nor Exponent Underflow shall be detected for the final results.

2.4.1.6 Exceptions

With respect to floating point exceptions, (specifically Exponent Overflow, Exponent Underflow, Indefinite, and Loss of Significance), Register assignments and bit position assignments within those Registers shall be in accordance with Section 2.8 of this specification.

NCR/CDC PRIVATE

a. When the use of standard floating point numbers generates an Out of Range result, the Exponent Overflow or Exponent Underflow exception shall correspondingly be indicated in the Condition Register

b. Whenever an Indefinite result is generated by a floating point operation, the corresponding Indefinite exception shall be indicated in the Condition Register.

c. When the use of standard floating point input arguments, by any floating point addition or subtraction instruction, would generate a final fraction result which would consist entirely of zeroes at the time the appropriate program interruption is enabled, the intermediate exponent and fraction results shall be used, unchanged, as the final results and the Loss of Significance exception shall be indicated in the Condition Register. For this set of circumstances, occurring at the time the appropriate program interruption is not enabled, the final result shall consist entirely of zeroes

2.4.1.7 Double Precision Register Designators

The terms "Xk+1" and "Xj+1" shall be used to designate an X Register associated with the rightmost part of a double precision floating point number. When the leftmost part of a double precision floating point number, as designated by the terms "Xk" and "Xj" is associated with Register XF (in hexadecimal notation) the terms "Xk+1" and "Xj+1" shall be interpreted as designating Register X0.

2.4.2 Conversion

The instructions within this subgroup shall provide the means for converting 64-bit words, contained in the X Registers, between floating point and integer formats.

2.4.2.1 Convert from Integer

Floating Point Convert from Integer, Floating Point (Xk) formed from Integer (Xj)
097 jk

NCR/CDC PRIVATE

This instruction shall convert the signed, two's complement, binary integer initially contained in the 64-bit positions of Register Xj to its equivalent, normalized floating point representation and shall transfer this 64-bit result to Register Xk. Integers outside of the range of -2^{48} through $2^{48}-1$ shall be truncated in the rightmost bit positions during conversion.

The integer initially contained in Register Xj shall be interpreted as having a magnitude (M) within the following range:

$$-2^{63} \leq M \leq 2^{63}-1$$

When the integer initially contained in Register Xj consists entirely of zeros, it shall be transferred without change to Register Xk.

2.4.2.2 Convert to Integer

Floating Point Convert to Integer, Integer (Xk) formed from Floating Point (Xj)
098 jk

This instruction shall convert the 64-bit floating point number initially contained in the Xj Register to a signed, two's complement, binary integer and shall transfer this 64-bit result to Register Xk. (The fractional part of the binary equivalent shall be lost as a result of truncation).

64-bit floating point numbers, initially contained in the Xj Register, shall be converted to 64-bit words consisting entirely of zeros whenever:

a. such numbers are Indefinite

b. such numbers have actual (unbiased) exponents which are less than or equal to zero

c. such numbers have co-efficients which consist entirely of zeros

Floating point numbers with magnitude (M) shall be correctly converted provided such numbers are within the following range:

$$-(2^{63}-2^{15}) \leq M \leq 2^{63}-2^{15}$$

NCR/CDC PRIVATE

For numbers outside of this range, the integer transferred to Register Xk shall represent only the least significant, (right-most) 64-bits of the actual result, an Overflow condition shall be detected, and if enabled the corresponding program interruption shall occur. (Thus, such numbers shall be truncated in their left-most bit positions).

2.4.3 Arithmetic

The instructions within this subgroup shall provide the means for performing arithmetic operations on floating point numbers to the extent described in the following subparagraphs.

2.4.3.1 Add/Subtract, Normalized/Unnormalized, Xk

a. Floating Point Sum, {Xk} replaced by {Xk} plus {Xj}
   099jk

b. Floating Point Difference, {Xk} replaced by {Xk} minus {Xj}
   100 jk

c. Floating Point Sum Unnormalized, {Xk} replaced by {Xk} plus {Xj}
   101 jk

d. Floating Point Difference Unnormalized, {Xk} replaced by {Xk} minus {Xj}
   102 jk

Operation: The subtract instructions in this subgroup shall differ from the add instructions in this subgroup only to the extent that they shall interpret the co-efficient sign initially contained in the Xj Register in its opposite state. Thus, conceptually, subtraction shall be accomplished by changing the sign of the subtrahend obtained from Register Xj and performing floating point addition.

These instructions shall add the 64-bit floating point number initially contained in the Xj Register {with the sign of the coefficient interpreted according to the operation code} to the 64-bit floating point number initially contained in the Xk Register, and shall transfer the 64-bit floating point result to the Xk Register as follows:

Exponent Equalization: Prior to aligning the coefficient by performing exponent equalization, the 48-bit coefficients from the floating point numbers initially contained in the Xk and Xj Registers shall be expanded to 96-bits each by appending 48 zeros to their rightmost bit positions. The exponents of the two floating point numbers, initially contained in the Xk and Xj Registers, shall be algebraically compared and when they are equal, that common exponent shall serve as the intermediate exponent result, with instruction execution proceeding directly to the coefficient arithmetic phase. However, when their exponents are not equal, the 96-bit expanded coefficient of the number associated

NCR/CDC PRIVATE

with the smaller exponent shall be shifted right, end-off, the number of bit positions designated by the difference between the exponents, up to a maximum of 96. Thus, the coefficients shall be aligned and the larger exponent shall be used as the intermediate exponent result.

Coefficient Arithmetic: The two aligned coefficients, each consisting of a signed fraction having 96-bits of precision, shall be algebraically added with the signed result consisting of an intermediate coefficient having 96-bits of precision and an overflow bit. {The overflow bit can be a one only when the coefficients have like signs}

Normalization: When the overflow bit resulting from the addition of the expanded and aligned coefficients is a one, the intermediate coefficient result shall be right shifted one bit position, end-off, with a one inserted on the left. This signed coefficient result, consisting of a normalized fraction having 96-bits of precision, shall be used in its leftmost 48-bit positions as the final coefficient. The intermediate exponent result shall increased by one, {to account for the right shift performed on the intermediate coefficient}, and the sum shall be used as the final exponent. The final signed coefficient result and the final biased exponent result shall be transferred to the 64-bit positions of the Xk Register, subject to the conditional treatment of exceptional results involving non-standard floating point numbers.

When the overflow bit resulting from the addition of the expanded and aligned coefficients is a zero, it shall be dropped from further consideration in accomplishing the floating point addition. The leftmost 48-bits of the intermediate coefficient result shall be used as the final result irrespective of the state of its leftmost bit or, shall be shifted left {up to a maximum of 47 bit positions with zeros inserted on the right} until its leftmost bit position is a one, as determined by the operation code. Thus, unnormalized arithmetic operations shall not involve any left shifts whatsoever and normalized arithmetic operations shall involve left shifts to the extent required to achieve normalization, with respect to the formation of the final coefficient result. For each bit position that the intermediate coefficient shall be shifted left, the intermediate exponent shall be decreased by one, unless the intermediate coefficient consists of 48 leading zeros, in which case both the intermediate exponent and intermediate coefficient shall be left unchanged. After left shifts to achieve normalization, along with the corresponding exponent adjustments, have been accomplished to the extent required and allowed, the coefficient sign, the

NCR/CDC PRIVATE

48 bits of the coefficient and the biased exponent shall be used as the final results and shall be transferred to the 64-bit positions of the Xk Register, subject to the conditional treatment of exceptional results involving non-standard floating point numbers.

Exceptions: These operations may result in Exponent Overflow, Exponent Underflow, Indefinite, or Significance Loss, conditions which shall be treated in the manner described in subparagraphs 2.4.1.3 and 2.4.1.6 with respect to the effects of these conditions on the floating point result and the generation of program interruptions, respectively, {Exponent Underflow conditions cannot occur for operation codes designating unnormalized addition or subtraction}.

Tables 2.4-2 and 2.4-3 indicate the generation of non-standard floating point results for combinations of both standard and non-standard floating point input arguments.

2.4.3.2 Product/Quotient, Xk

The following floating point multiply and divide instructions shall involve the prenormalization of the multiplier and multiplicand and the divisor and dividend, respectively. For each bit position that an input argument is shifted left, end off with a zero inserted on the right, the associated exponent shall be reduced by one, except when the coefficient consists entirely of zeroes in which case both the exponent and coefficient fields shall be left unchanged.

With respect to floating point exceptions, these operations may result in Exponent Overflow, Exponent Underflow or Indefinite conditions which shall be treated in the manner described in subparagraphs 2.4.1.3 and 2.4.1.6 with respect to the effects of these conditions on the floating point result and the generation of program interruptions. respectively.

When one or both input arguments for these instructions consist of non-standard floating point numbers, the non-standard floating point results shall be generated according to Tables 2.4-4 and 2.4-5 irrespective of prenormalization, exponent arithmetic, coefficient arithmetic or postnormalization operations.

a. Flt. Pt. Product, {Xk} replaced by {Xk} times {Xj}
103 jk

This instruction shall multiply the 64-bit floating point number

NCR / CDC PRIVATE

initially contained in Register Xj by the 64-bit floating point number initially contained in Register Xk and shall transfer the 64-bit floating point product to Register Xk.

Exponent Arithmetic: Following prenormalization, if required, the signed exponents of the multiplier and multiplicand shall be algebraically added and the sum shall be used as the intermediate exponent result.

Coefficient Arithmetic: The 48-bit normalized coefficient of the multiplicand shall be multipled by the 48-bit normalized coefficient of the multiplier and the result shall consist of an algebraically signed product having 96-bits of precision.

Postnormalization: When the leftmost bit of the 96-bit intermediate product is equal to a one, it shall be used in the leftmost 48-bit positions as the final coefficient result and the intermediate exponent, including bias, shall be used as the final exponent result.

When the leftmost bit of the 96-bit intermediate product is equal to a zero, the entire intermediate product shall be shifted left one bit position, end-off with a zero inserted on the right, and the intermediate exponent shall be reduced by one. The leftmost 48-bit positions of the shifted intermediate product shall be used as the final coefficient result and the reduced intermediate exponent, including bias, shall be used as the final exponent result.

The final 64-bit floating point result, consisting of the coefficient sign, the biased exponent, and the 48-bit fraction, shall be transferred to the Xk Register.

b. Flt. Pt. Quotient, {Xk} replaced by {Xk} divided by {Xj}
104 jk

This instruction shall divide the 64-bit floating point number initially contained in Register Xk by the 64-bit floating point number initially contained in Register Xj and shall transfer the 64-bit floating point quotient to Register Xk.

Exponent Arithmetic: Following prenormalization, if required, the signed exponent associated with the divisor shall be subtracted from the signed exponent associated with the dividend and the difference shall be used as the intermediate exponent result.

Coefficient Arithmetic: The 48-bit normalized coefficient of the

NCR / CDC PRIVATE

dividend shall be expanded to 96-bits by appending 48 zeroes to its rightmost bit position and these 96-bits shall be divided by the 48-bit normalized coefficient of the divisor. The quotient shall consist of an algebraically signed result having 48 bits of precision plus an overflow bit.

Postnormalization: When the overflow bit is a one the 48-bit quotient shall be shifted right one bit position, end-off with a one inserted on the left, and the intermediate exponent shall be increased by one. The shifted 48-bit quotient shall be used as the final coefficient result and the increased intermediate exponent, including bias, shall be used as the final exponent result.

When the overflow bit is equal to a zero, the 48-bit quotient shall be used as the final coefficient result and the intermediate exponent including bias, shall be used as the final exponent result.

Note: With respect to the formation of the quotient, the division of the coefficients shall generate an unrounded result according to the algorithm constraint as previously defined for integer divide in subparagraph 2.2.2.4 of this specification.

2.4.3.3 Add/Subtract, Xk and Xk+1

a. Flt. Pt. DP Sum, (Xk, Xk+1) replaced by (Xk, Xk+1) plus (Xj, Xj+1)
   105 jk

b. Flt. Pt. DP Difference, (Xk, Xk+1) replaced by (Xk, Xk+1) minus - (Xj, Xj+1)
   106 jk

Operation: These instructions shall add the double precision floating point number contained in Registers Xj and Xj+1 (with the sign of this coefficient interpreted according to the operation code; in its true state for Sum and in its inverted state for Difference) to the double precision floating point number initially contained in Registers Xk and Xk+1 and shall transfer the double precision floating point result to Registers Xk and Xk+1.

These instructions shall operate identically to the Normalized Sum and Normalized Difference instructions described in subparagraph 2.4.3.1 of this specification with the obvious exception that double precision rather than single precision floating point numbers shall be accomodated according to the format described in subparagrph 2.4.1.1 of this specification.

NCR/CDC PRIVATE

Only those phases of instruction execution differing from single precision operations shall be described and then, only to the extent of defining those differences.

Exponent Equalization: The 96-bit coefficients which shall be aligned during this phase of instruction execution shall be comprised in the rightmost 48-bit positions of the coefficient field from Register Xk+1 and Register Xj+1 with respect to the augend and addend, respectively.

Normalization: The 96-bit normalized coefficient, resulting from this phase of instruction execution, shall be used in its rightmost 48-bit positions as the fraction to be transferred to Register Xk+1. The coefficient sign transferred to Register Xk+1 shall be equal to the coefficient sign transferred to Register Xk.

When the execution of these instructions results in the generation of a standard floating point number, the biased exponent transferred to Register Xk+1 shall be 48 less than the biased exponent transferred to Register Xk.

When the execution of these instructions results in the generation of a non-standard floating point number, the 64-bit word transferred to Register Xk+1 shall be identical to the 64-bit word transferred to Register Xk.

2.4.3.4 Product/Quotient, Xk and Xk+1

Each of these double precision floating point instructions shall operate identically to the corresponding single precision floating point instruction described in subparagraph 2.4.3.2 with the obvious exception that double precision floating point numbers shall be accomodated according to the format described in subparagraph 2.4.1.1 of this specification.

Only those phases of instruction execution differing from single precision operations shall be described and then, only to the extent of defining those differences.

a. Flt. Pt. DP Product, (Xk, Xk+1) replaced by (Xk, Xk+1) times (Xj, Xj+1)
   107 jk

This instruction shall multiply the double precision floating point number initially contained in Registers Xj and Xj+1 by the double precision floating point number initially contained in Registers Xk and Xk+1 and shall transfer the double precision

NCR/CDC PRIVATE

floating point product to Registers Xk and Xk+1.

Coefficient Arithmetic: The multiplier and multiplicand shall each consist of a signed coefficient having 96 bits of precision. The result of the multiplication shall consist of an algebraically signed product having 192 bits of precision. The rightmost 95 bits of this product shall be truncated (discarded) and the leftmost 97-bits shall serve as the intermediate coefficient result to be normalized.

Postnormalization: The 96-bit normalized coefficient, resulting from this phase of instruction execution, shall be used in its rightmost 48-bit positions as the fraction to be transferred to Register Xk+1. The coefficient sign transferred to Register Xk+1 shall be equal to the coefficient sign transferred to Register Xk.

When the execution of this instruction results in the generation of a standard floating point number, the biased exponent tranferred to Register Xk+1 shall be 48 less than the biased exponent transferred to Register Xk. For non-standard results, see 2.4.1.3, item d.

b. Flt. Pt. DP Quotient, (Xk, Xk+1) replaced by (Xk, Xk+1) divided by (Xj, Xj+1)
108 jk

This instruction shall divide the double precision floating point number initially contained in Registers Xk and Xk+1 by the double precision floating point number initially contained in Registers Xj and Xj+1 and shall transfer the double precision floating point result to Register Xk and Xk+1.

Coefficient Arithmetic: The 96-bit normalized coefficient of the dividend shall be expanded to 192 bits by appending 96 zeroes to its rightmost bit position and these 192-bits shall be divided by the 96-bit normalized coefficient of the divisor. The quotient shall consist of an algebraically signed result having 96 bits of precision plus an overflow bit.

Postnormalization: The 96-bit normalized coefficient, resulting from this phase of instruction execution, shall be used in its rightmost 48-bit positions as the fraction to be transferred to Register Xk+1. The coefficient sign transferred to Register Xk+1 shall be equal to the coefficient sign tranferred to Register Xk.

When the execution of this instruction results in the generation of a standard floating point number, the biased exponent transferred to Register Xk+1 shall be 48 less than the biased exponent transferred to Register Xk. For non-standard results, see 2.4.1.3, item d.

NCR / CDC PRIVATE

		Xj			
		0	+∞	-∞	±IND
Xk	0	0	+∞	-∞	IND
	+∞	+∞	+∞	IND	IND
	-∞	-∞	IND	-∞	IND
	±IND	IND	IND	IND	IND

Table 2.4-2: (Xk) + (Xj)

		Xj			
		0	+∞	-∞	±IND
Xk	0	0	-∞	+∞	IND
	+∞	+∞	IND	+∞	IND
	-∞	-∞	-∞	IND	IND
	±IND	IND	IND	IND	IND

Table 2.4-3: (Xk) - (Xj)

		Xj						
		+N	-N	+0	-0	+∞	-∞	±IND
Xk	+N	+P	-P	0	0	+∞	-∞	IND
	-N	-P	+P	0	0	-∞	+∞	IND
	+0	0	0	0	0	IND	IND	IND
	-0	0	0	0	0	IND	IND	IND
	+∞	+∞	-∞	IND	IND	+∞	-∞	IND
	-∞	-∞	+∞	IND	IND	-∞	+∞	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

Table 2.4-4: (Xk) * (Xj)

Note: For the Key to symbols, see the following page.

NCR / CDC PRIVATE

		Xj						
		+N	-N	+0	-0	+∞	-∞	±IND
Xk	+N	+Q	-Q	+∞	-∞	0	0	IND
	-N	-Q	+Q	-∞	+∞	0	0	IND
	+0	0	0	IND	IND	0	0	IND
	-0	0	0	IND	IND	0	0	IND
	+∞	+∞	-∞	+∞	-∞	IND	IND	IND
	-∞	-∞	+∞	-∞	+∞	IND	IND	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

Table 2.4-5: {Xk} / {Xj}

		Xj						
		+N	-N	+0	-0	+∞	-∞	±IND
Xk	+N	D	<	<	<	>	<	IND
	-N	>	D	>	>	>	<	IND
	+0	>	<	=	=	>	<	IND
	-0	>	<	=	=	>	<	IND
	+∞	<	<	<	<	=	<	IND
	-∞	>	>	>	>	>	=	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

Table 2.4-6: Xj ≡ Xk

KEY:

For ±∞, ±IND and ±0, see Table 2.4-1

W = Any word except ±∞, ±IND
N = Any word except ±∞, ±IND, or ±0
S = Algebraic Sum
D = Algebraic Difference
P = Product
Q = Quotient

NCR/CDC PRIVATE

2.4.4 Branch

The instructions in this subgroup shall consist of conditional branch instructions.

Each of these conditional branch instructions shall perform a comparison between two floating point numbers. Then, based on the relationship between the results of that comparison and the branch condition as specified by means of the instruction's operation code, each conditional branch instruction shall perform either a normal exit or a branch exit.

Normal Exit: When the results of a comparison do not satisfy the branch condition as specified by the operation code, a normal exit shall be performed. A normal exit for all conditional branch instructions shall consist of adding four to the rightmost 32 bits of the PVA obtained from the P Register with that 32-bit sum returned to the P Register in its rightmost 32-bit positions.

Branch Exit: When the results of a comparison satisfy the branch condition as specified by the operation code, a branch exit shall be performed. A branch exit shall consist of expanding the 15-bit Q field from the instruction to 31 bits by means of sign extension, shifting these 31 bits left one bit position with a zero inserted on the right, and adding this 32-bit shifted result to the rightmost 32-bits of the PVA obtained from the P Register with the 32-bit sum returned to the P Register in its rightmost 32-bit positions.

2.4.4.1 Compare and Branch

Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} equal to {Xk}
109 jkQ

Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} not equal to {Xk}
110 jkQ

Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} greater than {Xk}
111 jkQ

Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} not less than {Xk}
112 jkQ

Operation: Each of these instructions shall perform an algebraic comparison of the 64-bit word obtained from Register Xj to the 64-bit word obtained from Register Xk. Each of these 64-bit words

NCR/CDC PRIVATE

shall be treated as a signed single precision floating point number as described in subparagraph 2.4.1.1 of this specification. The contents of Register X0 shall be interpreted as consisting entirely of zeroes.

When the coefficient signs are alike, a floating point subtract shall be performed in the manner described in subparagraph 2.4.3.1 of this specification, with the exception that the result shall not be transferred to Register Xk but shall be interpreted in its post-normalized form to determine the results of the comparision.

However, indefinite shall not be interpreted as being equal to, greater than, or less than any other number. With respect to non-standard floating numbers the results of comparisons shall occur according to Table 2.4-6.

These instructions shall perform a normal exit or a branch exit in the manner previously described in Paragraph 2.4.4 of this specification.

2.4.4.2 Exception Branch

Branch to {P} displaced by 2*@ if FLT.Pt.
Exception per j contained in Xk
113 jk@

This instruction shall perform a branch exit in the manner previously described in Paragraph 2.4.4 of this specification when the exception condition, as designated by the rightmost 2-bits of the j field from the instruction, is applicable to the 64-bit floating point number contained in the Xk Register.

This instruction shall perform a normal exit in the manner previously described in Paragraph 2.4.4 of this specification when the exception condition, as designated by the rightmost 2-bits of the j field from the instruction, is not applicable to the 64-bit floating point number contained in the Xk Register.

The values of the rightmost 2-bits of the j field from the instruction shall be associated with exception conditions as follows:

if 00, Exponent Overflow
if 01, Exponent Underflow
if 10 or 11, Indefinite

NCR/CDC PRIVATE

2.4.5 Compare

Compare Flt. Pt. {Xj} to {Xk}, result to X1 Right
114 jk

When either or both of the 64-bit floating point numbers contained in Register Xj and Register Xk are indefinite, the results of comparing these two numbers shall not be transferred to Register X1 Right and this instruction shall result in no operation.

The comparison of the 64-bit floating point numbers initially contained in Registers Xj and Xk shall be performed in a manner identical to that previously described in subparagraph 2.4.4.1 of this specification. For this instruction also, the contents of the X0 Register shall be interpreted as consisting entirely of zeroes.

When the initial contents of the Xj Register are equal to the initial contents of the Xk Register, Register X1 Right shall be cleared in all 32-bit positions.

When the initial contents of the Xj Register are greater than the initial contents of the Xk Register, Register X1 Right shall be cleared in the leftmost 31-bit positions, 32 through 62 and shall be set in the rightmost bit position 63.

When the initial contents of the Xj Register are less than the initial contents of the Xk Register, Register X1 Right shall be set in all 32 bit positions.

NCR/CDC PRIVATE

2.5 Logical Environment

A logical environment shall be defined by two sets of registers. The first set shall be referred to as the Processor State Register and shall include all items which are not unique to a process. Each processor shall have one set of Processor State Registers.

The second set of registers shall be referred to as the Process State Registers and shall include all items which are unique to a process. The act of going from one process state to another shall be referred to as an exchange. The contents of the Process State Registers associated with the exchange shall be referred to as an Exchange Package. Therefore, each process shall have one Exchange Package to define its unique environment.

2.5.1 Processor State Registers

See Table 2.5-1 and the following paragraphs for the definition of each of the Processor State Registers.

Processor State Register	Bit Positions (inclusive)
Job Process State	32 - 63
Monitor Process State	32 - 63
Page Table Address	32 - 63
Page Table Length	56 - 63
Page Size Mask	57 - 63
Processor Identification	40 - 63
System Interval Timer	32 - 63
One Megahertz Counter	0 - 63
Processor Test Mode	Processor model dependent
Processor Fault Status	Processor model dependent
Environment Control	Processor model-dependent

Table 2.5-1. Bit positions of Processor State Registers when copied to or from a 64-bit X Register.

NCR/CDC PRIVATE

2.5.1.1 Job Process State (JPS)

The JPS shall consist of a 32-bit real memory byte address. It shall point to the first entry in the exchange package for the job process. The JPS address shall be aligned with bits 00 through 31 of real memory addresses. The JPS address shall be 0, modulo 8. Bits 0, 29, 30, and 31 shall be ignored and treated as zeros.

Note: See 3.1.3 for the definition of a real memory address.

2.5.1.2 Monitor Process State (MPS)

The MPS shall consist of a 32-bit real memory byte address. It shall point to the first entry in the exchange package for the monitor process. The MPS address shall be aligned with bits 00 through 31 of real memory addresses. The MPS address shall be 0, modulo 8. Bits 0, 29, 30, and 31 shall be ignored and treated as zeros.

2.5.1.3 Page Table Address (PTA)

The PTA shall consist of a 32-bit real memory byte address. It shall point to the first entry in the Page Table. The PTA address shall be aligned with bits 00 through 31 of real memory addresses. The PTA address shall be 0, modulo the Page Table Length. Bit 0 and the rightmost bits defined as 0, modulo the Page Table Length, shall be ignored and treated as zeros.

NCR/CDC PRIVATE

2.5.1.4 Page Table Length (PTL)

The PTL shall consist of an 8-bit mask which shall specify the length of the Page Table. The PTL mask shall express the page table length in multiples of 4096 bytes. The mask shall consist of a contiguous string of one bits, beginning in the rightmost bit position of the PTL Register and extending towards the left-most bit position of the PTL Register. Thus, the number of 64-bit entries in the Page Table shall range from 512 (PTL Mask with all 8 bits clear) to 131,072 (PTL Mask with all 8 bits set). See paragraph 3.5.

2.5.1.5 Page Size Mask (PSM)

The PSM shall consist of a 7-bit mask which shall specify the page size used in allocating real central memory. The PSM shall express this page size in multiples of 512 bytes. The PSM shall consist of a contiguous string of one bits beginning in the left-most bit position of the PSM and extending towards the rightmost bit position of the PSM. Thus, the page size provided to a processor for its interpretation shall range from 64K bytes (PSM with all 7 bits clear) to 512 bytes (PSM with all 7 bits set). See subparagraph 3.4.2.2.

2.5.1.6 Processor Identification (PID)

The PID shall consist of 24-bits which shall uniquely identify each IPL processor, world-wide. The eight leftmost bits of the PID shall encode the processor's model number. The 16 rightmost bits of the PID shall encode the processor's serial number.

2.5.1.7 System Interval Timer (SIT)

The SIT shall be a 32-bit counter which the system shall use to establish a maximum time interval for job mode execution. See paragraph 2.5.3.3 for details.

2.5.1.8 One Megahertz Counter

The One Megahertz Counter shall consist of a free-running 64-bit counter. It shall increment once each microsecond. See paragraph 2.5.3.1 for details.

NCR/CDC PRIVATE

2.5.1.9 Processor Test Mode (PTM)

The PTM register shall provide the means for forcing faults within a processor in order to test its hardware fault-sensing logic. Moreover, the PTM shall provide the means for individually testing each fault-sensing mechanism within a Processor. Thus, the exact bit definitions of the PTM shall be model-dependent.

2.5.1.10 Processor Fault Status (PFS)

The PFS register shall provide the means for indicating a processor's hardware fault status. The exact bit definitions in the PFS shall be model-dependent.

2.5.1.11 Environment Control (EC)

The EC register shall provide the means for the Service Processor to control a processor's environment to the extent of determining its mode of operation. Except for the following, the bit definitions for the EC shall be model-dependent.

a. Monitor Mode: This bit, when set, shall place a processor in Monitor Mode. This bit shall be toggled during the execution of an Exchange operation.

b. Real Addressing Mode: This bit, when set, shall bypass the Virtual Addressing Mechanism such that all central memory references shall be performed with neither virtual address translation nor protection (validation), on the part of the associated processor.

c. Test Mode: This bit, when set, shall permit the copy instruction (reference number 131) to write into the Processor Test Mode and Fault Status Registers. Attempts to write into PTM or PFS when not in Test Mode shall result in an Instruction Specification Error.

NCR/CDC PRIVATE

2.5.2 Process State Registers

Each Process State shall be defined by an individual Exchange Package. An Exchange Package shall consist of a minimum of 52 64-bit words in Central Memory at contiguous word locations. The contents of an Exchange Package shall be formatted according to this specification such that corresponding interpretation by a processor shall provide the means for establishing a unique Process State.

Each Exchange Package in Central Memory shall contain Process State information in sufficient quantity and detail such that a processor may be dynamically switched between Exchange Packages. Moreover, when a processor is switched from a first Exchange Package to a second Exchange Package and at some later time is switched back to the first Exchange Package, the integrity of the processing which occurs for the Process State represented by the first Exchange Package shall not be affected.

Those items in the Exchange Package which shall exist in registers when an Exchange Package is active shall be processor model dependent. The processor model dependent specifications shall define those items.

Figure 2.5-2 defines the contents of the first 52 words in an Exchange Package. The sections which follow shall define the items contained in those words.

a. When the information contained in an Exchange Package is implicitly utilized in the course of instruction execution on the part of the associated "process" or is explicitly read, where applicable, by a "Copy to Xk per (Xj)" instruction (reference No. 130), the states of the following bits shall be ignored and treated as zeros.

Word 0:	Bits 00, 01, 08, 09 and 63	(P)
Word 2:	Bits 02 through 13	(Unused)
Word 7:	Bits 00 through 03	(Unused)
Word 16:	Bits 00 through 03	(Unused)
Word 35:	Bits 13, 14, and 15	(Unused)
Word 36:	Bits 08, 09, 10 and 61 through 63	(Unused and DLP)

NCR/CDC PRIVATE

When the information contained in an Exchange Package is implicitly updated in the course of instruction execution on the part of the associated "process" or is explicitly written, where applicable, by a "Copy from Xk per (Xj)" instruction (reference number 131), the states of these same bits shall be undefined.

b. The statements made in item a. shall also apply to the Exchange Package, Word 3, Bits 00 through 06 (leftmost 7-bit positions of the User Mask) with the exception that these bits shall be treated as ones.

c. When the information contained in words 37 through 51 of an Exchange Package is utilized during "call," "return," "pop" or "exchange" operations on the part of the associated "process," bits 0 through 15 of these words shall not be altered in central memory.

d. The modification of Process State Register values in a central memory exchange package by one processor at the time that process is being executed by another processor, shall result in undefined operation. Overlapped exchange packages in central memory may also result in undefined operations.

e. To provide the alternative, on a model-dependent basis, for cache addressing by means of PVA or SVA, exchange packages should be kept in Cache By-Pass Segments in order to prevent any anomalous operations which might result from "stale" cache data conditions.

NCR/CDC PRIVATE

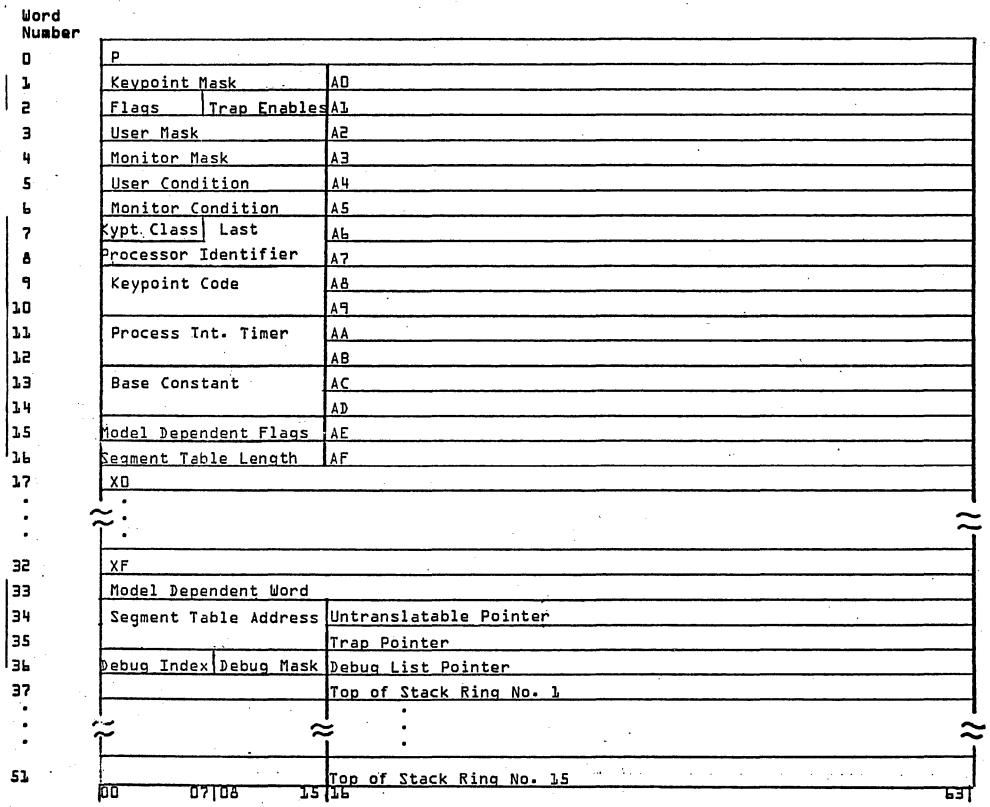


Figure 2.5-2
Exchange Package

NCR/CDC PRIVATE

2.5.2.1 Program Address Register (P)

See paragraph 2.1.1.1 for the definition of the P Register's contents.

P shall be located in bits 00 through 63 of word 0 in the Exchange Package.

2.5.2.2 A Registers

The 16 A Registers, A0 through AF, shall be located in bits 16 through 63 of words 1 through 16, respectively, in the Exchange Package. See paragraph 2.1.1.2. for the definition of the A Register's contents.

2.5.2.3 X Registers

The 16 X Registers, X0 through XF shall be located in bits 00 through 63 of words 17 through 32, respectively, in the Exchange Package. See paragraph 2.1.1.3 for the definition of the X Register's contents.

2.5.2.4 Kypt Class (Keypoint Class Number, KCN)

KCN shall consist of a 4-bit code which identifies the keypoint class number which resulted in a keypoint interrupt. The KCN recorded shall correspond to the keypoint mask bit number that enabled the keypoint interrupt. See 2.5.2.11 and 2.6.1.7.

The KCN shall be located in bits 04 through 07 of word 7 in the Exchange Package.

2.5.2.5 Flags

The Flags field shall consist of two separate single bit flags which have the following definitions. The Copy instructions (reference numbers 130 and 131) shall provide the means for setting and clearing these flags on the part of the associated process.

a. Critical Frame Flag (CFF).
The CFF, if set, shall indicate that the currently active stack frame for the process defined by this Exchange Package is a "critical frame." In this context, software shall have exclusive control over the state of CFF.

CFF shall be located in bit 0 of word 2 in the Exchange Package.

b. On Condition Flag (OCF)

The OCF is intended to facilitate the handling of "on condition" traps on the part of the "process monitor." In this context, software shall have exclusive control over the state of OCF.

NCR/CDC PRIVATE

OCF shall be located in bit 1 of word 2 in the Exchange Package.

2.5.2.6 User Mask (UM)

UM shall be used by user processes to enable trap interrupts. There shall be 16 bits in the UM. See paragraph 2.8.4 for details.

The UM shall be located in bits 00 through 15 of word 3 in the Exchange Package.

2.5.2.7 Monitor Mask (MM)

MM shall be used by the monitor to enable exchange interrupts. There shall be 16 bits in the MM. See paragraph 2.8.2 for details.

The MM shall be located in bits 00 through 15 of word 4 in the Exchange Package.

2.5.2.8 User Condition Register (UCR)

UCR shall be a 16-bit register which records the occurrence of specified conditions within the processor. See paragraph 2.8.3 for details.

UCR shall be located in bits 00 through 15 of word 5 in the Exchange Package.

2.5.2.9 Monitor Condition Register (MCR)

MCR shall be a 16-bit register which records the occurrence of specified conditions within the processor and central memory. See paragraph 2.8.1 for details.

MCR shall be located in bits 00 through 15 of word 6 in the Exchange Package.

2.5.2.10 Debug Mask (DM)

DM shall be used to help debug programs by enabling trap interrupts to occur for any of several reasons. Section 2.7.2 describes the debug operation.

The DM bits shall be located in bits 11 through 15 of word 36 in the Exchange Package. The assignments are:

bit 11 - Data Read, first address of string

bit 12 - Data Write, first address of string

bit 13 - Instruction fetch

bit 14 - Branching instruction

bit 15 - Call instruction

2.5.2.11 Keypoint Mask (KM)

KM shall consist of a 16-bit mask which selectively enables keypoint interrupt to occur whenever a keypoint instruction is executed with a keypoint code which corresponds to the KM. See paragraph 2.7.1.1.

The KM shall be located in bits 00 through 15 of word 1 in the Exchange Package.

2.5.2.12 Keypoint Code (KC)

KC shall consist of a 32-bit code which defines the keypoint event which caused the keypoint interrupt. See paragraph 2.7.1.1 and the Keypoint instruction (reference number 136.)

The KC shall be located in bits 00 through 15 of words 9 and 10 in the Exchange Package. Word 9 shall contain the leftmost 16 bits of the KC.

2.5.2.13 Process Interval Timer (PIT)

PIT shall be a 32-bit counter which a process shall use to determine time intervals. See paragraph 2.5.3.2 for details.

The PIT shall be located in bits 0 through 15 of words 11 and 12 in the Exchange Package. Word 11 shall contain the leftmost 16 bits of PIT.

2.5.2.14 Base Constant (BC)

The BC is intended to provide the means for communication from "System Monitor" to "Process Monitor" by designating the BN field of the Segment 0 PVA which points to the corresponding process' "Run Message In Buffer." See OS GDS, Section IX; Run Descriptors.

The BC shall be located in bits 00 through 15 of words 13 and 14 in the Exchange Package. Word 13 shall contain the leftmost 16 bits of BC.

2.5.2.15 Model Dependent Flags (MDF)

MDF shall consist of 16 bits. MDF shall be processor model dependent and shall be defined in the processor model dependent specification.

MDF shall be located in bits 00 through 15 of word 15 in the Exchange Package.

2.5.2.16 Segment Table Length (STL)

STL shall specify the number of entries in the Segment Table.

It shall be used to verify that references to the Segment Table are actually within the defined Segment Table. STL shall be a positive 12-bit value. See paragraph 3.3.

The STL shall be located in bits 4 through 15 of word 16 in the Exchange Package.

2.5.2.17 Untranslatable Pointer {UP}

UP shall be the PVA which caused an Interrupt because it could not be translated into a real address.

The UP shall be located in bits 16 through 63 of word 34 in the Exchange Package.

2.5.2.18 Segment Table Address {STA}

STA shall be a real memory byte address that points to the first entry in the Segment Table. See paragraph 3.3. STA shall be interpreted as equal to 0, modulo 8. STA shall be located in bits 00 through 15 of words 34 and 35 of the Exchange Package. Word 34 shall contain the leftmost 16 bits of STA.

2.5.2.19 Last Processor Identification {LPI}

LPI shall be a 24-bit identifier that identifies the last processor to execute the process defined by the current Exchange Package. Bits 08 through 15 shall contain the model number and bits 00 through 15 shall contain the serial number of the last processor in words 7 and 8, respectively, of the Exchange Package.

2.5.2.20 Trap Enables {TE}

TE shall consist of a 2-bit field that determines how traps shall be enabled. The bits in TE shall be set by the "Copy from Xk per {Xj}" instruction {reference number 1311}. Although the bits in TE can be cleared by the "Copy from Xk per {Xj}" instruction they shall normally be cleared by the hardware action described below. See section 2.8.6 for a description of trap interrupt operation.

a. Trap Enable Flip-flop {TEF}

TEF shall be the flip-flop which enables a trap interrupt operation to occur when it is set. It shall be set as described above and shall be cleared by hardware whenever a trap interrupt occurs.

TEF shall be located in bit 14 of word 2 in the Exchange Package.

b. Trap Enabled Delay {TED}

TED shall be a flip-flop which delays the enabling of trap interrupts until after the next Return instruction {reference number 117} is executed. The trap enable shall be inhibited as long as TED is set. The Return instruction clears TED. TED shall be set by the Copy instruction as just previously described.

TED shall be located in bit 15 of word 2 in the Exchange Package.

NCR/CDC PRIVATE

DOCUMENT | SECTION
NCR | CONTROL DATA | THIS | REPLACES
ADVANCED SYSTEMS LABORATORY | FILE | | DATE Dec. 9, 1974 | Sept. 30, 1974
APPROVED ASL | APPROVED NCR | APPROVED CDC | PAGE 124 | 124

2.5.2.21 Trap Pointer {TP}

TP shall consist of a PVA which points to a code base pointer in a binding section. The TP shall be used whenever a trap interrupt occurs. See 2.8.6.

The TP shall be located in bits 16 through 63 of word 35 in the Exchange Package.

2.5.2.22 Debug Index {DI}

DI shall consist of 2 flags and a 6-bit word-index into the debug list. It shall record where the debug list search must resume after a debug list find has been processed. See. 2.7.2.3.

The DI shall be located in bits 00 through 07 of word 36 in the Exchange Package.

2.5.2.23 Debug List Pointer {DLP}

DLP shall consist of a PVA that points to the first entry in the debug list. See 2.7.2.1.

The DLP shall be located in bits 16 through 63 of word 36 in the Exchange Package.

2.5.2.24 Top of Stack {TOS}

Each TOS shall consist of a PVA that points to the top of its associated stack. There shall be an individual TOS pointer for each of the 15 rings.

The TOS's shall be located in bits 16 through 63 of words 37 through 51 in the Exchange Package. The TOS for ring 1 shall be located in word 37, the TOS for ring 2 shall be located in word 38, etc.

2.5.2.25 Model Dependent Word {MDW}

MDW shall consist of 64-bits, shall be processor model dependent and shall be defined in the processor model dependent specification.

MDR shall be located in bits 00 through 63 of word 33 in the Exchange Package.

NCR/CDC PRIVATE

2.5.3 Clocks

2.5.3.1 One Megahertz Counter

The One Megahertz Counter shall be a free-running 64-bit counter that shall increment once each micro second. It shall be cleared by a power-on master clear and shall run as long as power is on. The Copy instructions (reference numbers 130 and 131) shall provide the means for reading and writing the One Megahertz Counter.

2.5.3.2 Process Interval Timer

The Process Interval Timer (PIT) shall be a 32-bit counter that shall decrement once each microsecond. When it decrements to zero it shall set the Process Interval Timer bit in the Condition Register. This, in turn, shall cause a trap interrupt to occur per the masking and enabling requirements described in section 2.8.

The PIT contains a different count for each User process. When a particular process is not in active execution its PIT value is stored in its Exchange Package. By this means each User process may keep track of time intervals within its own program execution.

PIT shall be set by the "Copy from Xk per (Xj)" instruction described in section 2.6.5.2, as well as during an "Exchange" operation, described in 2.6.1.6 and 2.8.5.

2.5.3.3 System Interval Timer

The System Interval Timer (SIT) shall be a 32-bit counter that shall decrement once each microsecond. When it decrements to zero it shall set the System Interval Timer bit in the Condition Register. This, in turn, shall cause an interrupt to occur per the masking and enabling requirements described in section 2.8.

By this means the Monitor process may keep track of time intervals within the processor. SIT shall be set by the "Copy from Xk per (Xj)" instruction described in section 2.6.5.2.

NCR/CDC PRIVATE

2.5.4 Stacks

Each process shall have the means for addressing 15 stacks, one for each possible ring of execution as determined by the value of the ring number contained in the P Register.

The beginning of each stack shall be defined by the PVA referred to as the Top of Stack pointer, previously described in subparagraph 2.5.2.24 and illustrated in Figure 2.5-2 of this specification.

Note: TOS pointers shall be addressed, using real addressing mode, as follows:

Address of TOS pointer = (Job Process State Register or Monitor Process State Register) plus (288) plus (8 times the value of the ring number contained in the P Register.)

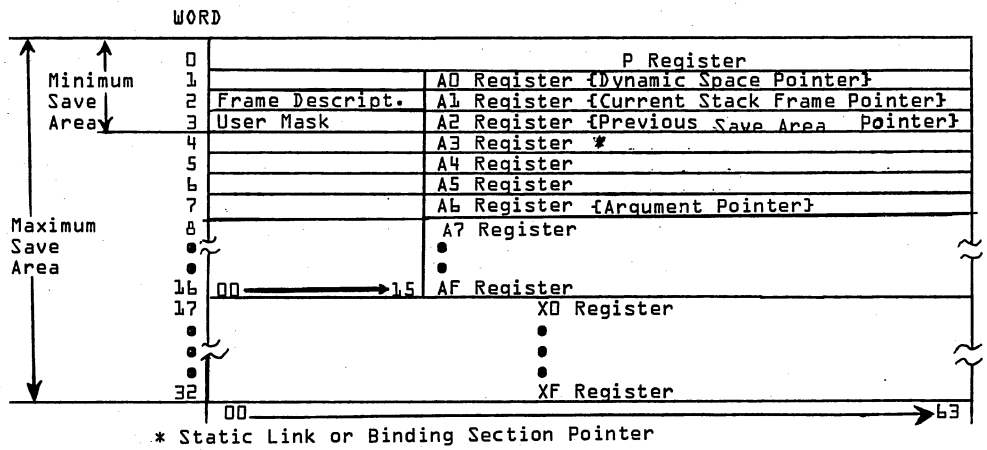
2.5.4.1 Stack Frames

Each stack shall be comprised of one or more stack frames. The beginning of each stack frame shall be defined by the PVA referred to as the Current Stack Frame Pointer. At the time a procedure is activated (or called) the CSF pointer shall be obtained by using the TOS pointer which corresponds to the procedure's ring of execution. During the time a procedure utilizes a stack frame, its length, from the beginning address, shall be defined as including each contiguous PVA up to, (but not including), the PVA referred to as the Dynamic Space Pointer.
When within a process, a procedure "calls" another procedure, with the intention that the "called" procedure will "return" to its "caller," the stack frame associated with the "calling" procedure is intended to provide the means for preserving its environment so that its execution may be suspended, (at the time the other procedure is "called"), and then resumed, (at the time the "called" procedure "returns").

At the end of each stack frame, a "save area" shall be defined for that part of a procedure's "environment" which is implicit to the Call and Return instructions as defined in subparagraphs 2.6.1.2 through 2.6.1.4 of this specification. The stack frame save area shall consist of from four to thirty-three contiguous 64-bit words, beginning at the address defined by the Dynamic Space Pointer with respect to Call instructions and beginning at the address defined by the Previous Save Area Pointer with respect to the Return instruction.

NCR/CDC PRIVATE

The Stack Frame Save Area shall be formatted as follows:



* Static Link or Binding Section Pointer

Thus the "environment" which is implicit to the Call and Return instructions shall include:

Minimally; P Register
Register A0 through A2 {DSP, CSF and PSA}
Frame Descript. {Stack Frame Save Area Descriptor}
User Mask

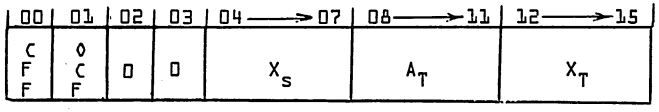
Selectively; Register A3 through AF {contiguously numbered}
Register X0 through XF {contiguously numbered}

Notes: The PVA initially contained in the P Register shall be increased by four prior to writing the entire P Register {including its Global and Local Key fields}, into the Current Stack Frame Save Area, Word 0, whenever such an operation occurs on the part of a Call instruction's execution.

Unused fields in the Stack Frame Save Area shall be cleared during the execution of Call instructions and shall be ignored during the execution of a Return instruction.

NCR/CDC PRIVATE

The Stack Frame Save Area Descriptor shall consist of 16-bits formatted as follows:



CFF: Critical Frame Flag
0CF: On Condition Flag
X_S : X Register, starting number {First X Reg. No.}
A_T : A Register, terminating number {Last A Reg. No.}
X_T : X Register, terminating number {Last X Reg. No.}

Trap Interrupt shall generate a maximum Stack Frame Save Area {33 words}, by definition.

For Call instructions, the A and X Registers to be stored into the Stack Frame Save Area shall be interpreted according to the contents of Register X0 Right, in the manner described in subparagraph 2.2.1.7 of this specification, with the exception that bit positions 48 through 51 of Register X0 Right shall be ignored and the storing of the A Register group shall unconditionally begin with Register A0. When X_S is greater than X_T, none of the X Registers shall be stored by Call instructions, and none shall be loaded by a Return instruction.

The execution of a Call instruction or a Trap Interrupt shall store the states of the Critical Frame and On Condition Flags into the Frame Descriptor associated with the Stack Frame Save Area. The execution of a Return instruction shall load these Flags from the Frame Descriptor contained within the Stack Frame Save Area.

NCR/CDC PRIVATE

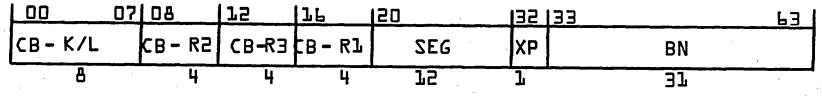
2.5.5 Binding Section Segment

A Binding Section Segment shall be identified by the RP field within its associated Segment Descriptor as described in 3.3.1.1 of this specification.

Binding Section Segments are intended to facilitate software linking of both code and data segments from one procedure to another.

Also, the detection of a PVA having a ring number of zero, along with the corresponding program interruption, as described for the Load instructions in subparagraph 2.2.1.6 of this specification, are performed for the purpose of identifying "unlinked" pointers.

With respect to the Call instruction, as described in subparagraph 2.6.1.2 of this specification, having both inter-ring and inter-segment branching capabilities, a Binding Section Segment shall be used to contain the Code Base Pointer to the "called" procedure. The Code Base Pointer shall be located on a word boundary, shall consist of 64-bits and shall have the following format:



With respect to the "called" procedure these fields shall have the following interpretation:

CB-K/L : Code Base Pointer, Global Lock and Local Key/Lock
CB-R2 : Code Base Pointer, Highest Ring Number for Execute
CB-R3 : Code Base Pointer, Highest Ring Number for Call
CB-R1 : Code Base Pointer, Lowest Ring Number for Execute
SEG - Segment Number
BN - Byte Number

Note: When the External Procedure Flag is a one, the next contiguous word location from the Code Base Pointer shall contain a PVA in its rightmost 48-bit positions, 16 through 63, referred to as a Binding Section Pointer. Thus, a new Binding Section Pointer shall be provided at the address of the Code Base Pointer plus 8 when an "external procedure" is "called".

NCR/CDC PRIVATE

2.6 System Instructions

2.6.1 Non-privileged Mode

This class of instructions shall be permitted to execute in any processor mode.

2.6.1.1 Execute Algorithm and Program Error

a. Execute Algorithm
139Sjkℓ

This instruction shall be a processor model dependent instruction. As such, it shall be defined in the appropriate processor model dependent specification.

S field	Use	Defining Document
0	Century Compatibility	P1 Spec; ASL00210
1		
2		
3		
4		
5		
6		
7		

Note: For those processors in which one or more of the above algorithms have not been implemented, the corresponding Execute Algorithm instructions shall result in the recording of an Unimplemented Instruction condition.

b. Program Error
121 jk

The execution of this instruction shall result in the detection of an Instruction Specification error and the corresponding program interruption shall occur.

The operation code for this instruction shall consist entirely of zeroes.

The j and k fields from this instruction shall not be translated and their values shall have no effect on the execution of this instruction.

NCR/CDC PRIVATE

2.6.1.2 Call per {Ai} displaced by 8*D, Arguments per {Aj}, Static Link per {Ak}

115 jkiD

Operation. This instruction shall save the "environment", as designated by the contents of Register X0 Right, in the stack frame save area pointed to by the Dynamic Space Pointer initially contained in Register A0. The stack associated with the current ring of execution, as determined by the RN field initially contained in the P Register, shall be "pushed" by transferring the Dynamic Space Pointer, modified in its rightmost 32-bit positions by the addition of 8 times the number of words stored into the stack frame save area, to the appropriate Top of Stack entry in the executing process' Exchange Package.
The PVA obtained from Register Ai shall be modified in its rightmost 32-bit positions by the addition of the zero-extended D field from the instruction, {shifted left 3-bit positions with zeroes inserted on the right}, and the resulting PVA shall be used to address a Code Base Pointer from a Binding Section Segment. This Code Base Pointer shall be translated into a PVA used to address the first instruction to be executed in the "called" procedure. The ring of execution of the called procedure, P{RN} final, shall be used to obtain a Top of Stack pointer from the process' Exchange Package to be used as the new Current Stack Frame Pointer.

The A0, A1, and A2 Registers shall be altered to reflect changes with respect to the Current and Previous Stack Frames and the A3, and A6 Registers shall be altered to reflect pertinent parameter changes as required, in accomplishing this transfer of control from a "calling" procedure to a "called" procedure.

Register assignments shall be as follows:

{A0} - Dynamic Space Pointer
{A1} - Current Stack Frame Pointer
{A2} - Previous Save Area Pointer
{A3} - Static Link or Binding Section Pointer
{A6} - Argument Pointer

Note: Execute Access Validation shall be performed by using the Code Base Pointer {as previously described in paragraph 2.5.5 of this specification} instead of the Segment Descriptor associated with the SEG field contained in the Code Base Pointer.

NCR/CDC PRIVATE

The associated program interruption shall occur and the execution of this instruction shall be inhibited when any of the exceptions are recorded from the following sequence of exception sensing:

Instruction Specification error when the value of the 4-bits in bit positions 56 through 59 of Register X0 Right is less than 2.

Address Specification Error
{Ai} not equal to 2, modulo 8.
An invalid PVA {bit position 32 equal to a one} for any access to the Binding Section.

Segment Descriptor Invalid for the Binding Section Segment.

Access Violation {See 3.3.1.1 and 3.6.2.2}
Code Base Pointer not contained in a "Binding Section" Segment
Code Base Pointer not contained in a "ring-readable" Segment

Page Table Search Without Find for the Binding Section page{s}

Ring Number Zero when the Code Base Pointer R1 is equal to zero.

Access Violation
Ai Ring Number greater than Code Base Pointer R3
Initial P Ring Number greater than Code Base Pointer R3
Initial P Global Key not equal to Code Base Pointer Global Lock in the absence of "master key" and "no-lock" values, respectively.

Outward Call when the initial P Ring Number is less than the Code Base Pointer R1.

Address Specification Error
Final {P} would not be equal to 0, modulo 2.
An invalid PVA {bit position 32 equal to a one} for any access to the Current Stack Frame Save Area.
Initial {A0} not equal to 0, modulo 8.

Segment Descriptor Invalid for the Current Stack Frame Save Area.

Access Violation
Current Stack Frame Save Area not in a "writable" Segment {See 3.3.1.1, 3.6.2.2 and 3.6.3.2}

Page Table Search Without Find for the Current Stack Frame Save Area page{s}.

NCR/CDC PRIVATE

In the absence of a program interruption, the following sequence of events shall accomplish the execution of the instruction:

	Operation		Remarks
*	a.	"Environment" to Stack Frame Save Area, then Copy P Left to X0 Left	See paragraph 2.5.4.1 Copy Caller's ID
*	b.	Store initial {A0}, incremented by 8 times the number of save area words, to Exchange Package per initial P Ring Number	Update TOS pointer See paragraph 2.5.2.24
*	c.	If P Global Key is not a "Master Key" go to step e.	See subparagraph 3.6.3.2 and paragraph 2.5.5
*	d.	Load P Global Key with Code Base Pointer Global Lock	
*	e.	Load P Local Key with Code Base Pointer Local Lock	
*	f.	If P Ring Number is not greater than Code Base Pointer R2, go to step h.	Intra-ring Call
*	g.	Set P Ring Number equal to Code Base Pointer R2	Inward Call
*	h.	Load P SEG and BN fields with Code Base Pointer SEG and BN fields	Clear bit position 32 of P Register
**	i.	If Code Base Pointer "XP" is 0, Copy {Ak} to A3 and go to step k	Internal Procedure
*	j.	Load A3 with new Binding Section Pointer	See paragraph 2.5.5
**	k.	Copy {Aj} to A6	Pass parameters
*	l.	Copy initial {A0} to A2	DSP to PSA pointer
*	m.	Clear On Condition Flag	Clear OCF
*	n.	Load A1 from Exchange Package per final P Ring Number and clear Critical Frame Flag	TOS to CSF pointer Clear CCF
*	o.	If {A1} equal to 0, modulo 8, go to step q.	
*	p.	{A1} + 7 to A1, 0 modulo 8 result.	Round upwards; word address
*	q.	Copy A1 to A0	CSF pointer to DSP

* Unconditionally included in a Trap Interrupt;
** Unconditionally omitted from a Trap Interrupt;
See 2.8.6

NCR/CDC PRIVATE

2.6.1.3 Call to {P} displaced by 2*Q, Arguments per {Aj}, Static Link per {Ak}.

116 jkQ

Operation. This instruction shall save the "environment", as designated by the contents of Register X0 Right, in the stack frame save area pointed to by the Dynamic Space Pointer initially contained in Register A0. The stack associated with the current ring of execution, as determined by the RN field initially contained in the P Register, shall be "pushed" by transferring the Dynamic Space Pointer, modified in its rightmost 32-bit positions by the addition of 8 times the number of words stored into the stack frame save area, to the appropriate Top of Stack entry in the executing process' Exchange Package.

The P Register shall be modified in its rightmost 32-bit positions by the sign extended Q field from the instruction, {left shifted 1-bit position with a zero inserted on the right} and the final contents of the P Register shall be used to address the first instruction to be executed in the "called" procedure.

Registers A0, A1 and A2 shall be altered to reflect changes with respect to the Current and Previous Stack Frames and the A3 and A6 Registers shall be altered to reflect pertinent parameter changes as required, in accomplishing this intra-ring, intra-segment transfer of control from a "calling" procedure to a "called" procedure.

Register assignments shall be as follows:

{A0} - Dynamic Space Pointer
{A1} - Current Stack Frame Pointer
{A2} - Previous Save Area Pointer
{A3} - Static Link
{A6} - Argument Pointer

NCR/CDC PRIVATE

The associated program interruption shall occur and the execution of this instruction shall be inhibited when any of the exceptions are recorded from the following sequence of exception sensing:

Instruction Specification error when the value of the 4-bits in positions 56 through 59 of Register X0 Right is less than 2.

Address Specification Error
Initial {A0}not equal to 0, modulo 8.

An invalid PVA {bit position 32 equal to a one} for any access to the Stack Frame Save Area.

Segment Descriptor Invalid for the Stack Frame Save Area Segment.

Access Violation {See 3.3.1.1, 3.6.2.2 and 3.6.3.2}
Current Stack Frame Save Area not in a "writable" Segment

Page Table Search without Find for the Stack Frame Save Area page{s}.

In the absence of a program interruption, the following sequence of events shall accomplish the execution of this instruction:

	Operation	Remark
a.}	"Environment" to Stack Frame Save Area, then Copy P Left to X0 Left	See paragraph 2.5.4.1 Copy Caller's ID
b.}	Store initial {A0}, incremented by 8 times the number of save area words, to Exchange Package per initial P Ring Number	Update TOS pointer See paragraph 2.5.2.24
c.}	{P} plus 2*0 to P	Intra-ring, intra-segment Call
d.}	Copy {Ak} to A3 and {Aj} to A6	Pass parameters
e.}	Copy initial {A0} to A2 and clear Critical Fence Flag	DSP to PSA pointer Clear CFF
f.}	Copy initial {A0}, incremented by 8 times the number of save area words, to A0 and A1.	TOS to CSF pointer
g.}	Clear On Condition Flag	Clear OCF

NCR/CDC PRIVATE

2.6.1.4 Return
117jk

Operation. This instruction shall re-establish the Stack Frame and "environment" of a previous procedure as defined by the Previous Save Area Pointer.

The j and k fields from this instruction shall not be translated. Thus, their values shall have no effect on the execution of this instruction for which all execution parameters shall be implicit.

The Stack Frame Save Area from which a previous procedure's "environment" shall be obtained, shall be addressed by means of the PVA initially contained in Register A2. The format of the previous procedure's Stack Frame Save Area shall conform to the description contained in paragraph 2.5.4.1 of this specification.

The associated program interruption shall occur and the execution of this instruction shall be inhibited when any of the exceptions are recorded from the following sequence of exception sensing:

Address Specification Error when the initial {A2} not equal to 0, modulo 8, or an invalid PVA, {bit 32 equal to one}.

Segment Descriptor Invalid with respect to the PVA initially contained in Register A2.

Access Violation when initial {A2} would not address a "readable" segment {See 3.3.1.1, 3.6.2.2 and 3.6.3.2}

Page Table Search Without Find with respect to any central memory accesses to the previous procedure's Stack Frame Save Area.

Environment Specification Error
The value of the field designating the last A Register to be loaded, as contained in the Previous Stack Frame's Descriptor, is less than 2.

Segment Descriptor Invalid with respect to the PVA contained in Word 0 of the previous procedure's Stack Frame Save Area.

Address Specification Error
Final {P} would not be equal to 0, modulo 2.
Final {P} would be an invalid PVA, {bit 32 equal to one}.

NCR/CDC PRIVATE

Access Violation
Final {P} would not address an "executable" segment.
Final {P} Local Key would not "strictly - equal" the associated segment's Local Lock.
Final {P} Global Key would not "strictly - equal" the associated segment's Global Lock, provided the associated segment's Global Lock is not a "No Lock."

Note: The term "strictly - equal" infers bit-for-bit equivalence.

Environment Specification Error
Final P ring number would be less than initial A2 ring number.
Final {A0} would not equal initial {A2}.

Outward Call {Inward Return} if initial P ring number is greater than initial A2 ring number.

Critical Frame Flag if the initial state of the Critical Frame Flag is equal to a one.

In the absence of a program interruption, the following sequence of events shall accomplish the execution of this instruction:

Operation		Remarks
a.	Load the "environment "from the previous procedure's Stack Frame Save Area.	See 2.5.4.1 {Loaded per Frame Descriptor X_S, A_T and X_T; A_S=0}
b.	For each A Register ring number which is less than the P Register's ring number, set the associated A Register's ring number equal to the P Register's ring number.	Ripple
c.	Store the final {A1} to the Exchange Package per the final P ring number.	CSF→TOS pointer See paragraph 2.5.2.24
d.	Clear the Trap Enable Delay if set.	See 2.5.2.20

2.6.1.5 Pop
118 jk

NCR/CDC PRIVATE

Operation. This instruction shall re-establish the Stack Frame of a previous procedure as defined by the Previous Stack Frame's Save Area.

The j and k fields from this instruction shall not be translated. Thus, their values shall have no effect on the execution of this instruction for which all execution parameters shall be implicit.

The Stack Frame Save Area from which a previous procedure's Stack Frame pointers shall be obtained, shall be addressed by means of the PVA initially contained in Register A2. The format of the previous procedure's Stack Frame Save Area shall conform to the description contained in paragraph 2.5.4.1 of this specification.

The associated program interruption shall occur and the execution of this instruction shall be inhibited when any of the exceptions are recorded from the following sequence of exception sensing:

Address Specification Error
Initial {A2} not equal to 0, modulo 8.
Initial {A2} an invalid PVA {bit 32 equal to one}.

Segment Descriptor Invalid with respect to the Segment Descriptor associated with the PVA initially contained in Register A2.

Access Violation {See 3.3.1.1, 3.6.2.2 and 3.6.3.2}
Initial {A2} does not address a "readable" segment.

Page Table Search Without Find with respect to the central memory accesses to the previous procedure's Stack Frame Save Area.

Environment Specification Error
Initial {A2} not equal to Word 1 contained in the previous procedure's Stack Frame Save Area.

Inter-Ring Pop if the RN field contained in the P Register is not equal to the RN field initially contained in Register A2.

Critical Frame Flag if the initial state of the Critical Frame Flag is equal to a one.

In the absence of a program interruption, the following sequence of events shall accomplish the execution of this instruction.

Operation		Remarks
a.	Load A1 with the PVA from Word 2 of the previous procedure's Stack Frame Save Area	Update CSF pointer

NCR/CDC PRIVATE

b.) Load A2 with the PVA from Word 3 of the previous procedure's Stack Frame Save Area. — Update PSA pointer

c.) Load the Critical Fence Flag and the On Condition Flag from the previous procedure's Stack Frame Save Area. — Update CFF and OCF

d.) Store the final (A1) to the process' Exchange Package per the P Register's Ring Number — Update TOS pointer.

TOS = Top of Stack pointer
n = Ring of execution, inner ring
n+ = Ring of execution, outer ring
A0 = DSP = Dynamic Space pointer
A1 = CSF = Current Stack Frame pointer
A2 = PSA = Previous Save Area pointer
* = Save Area

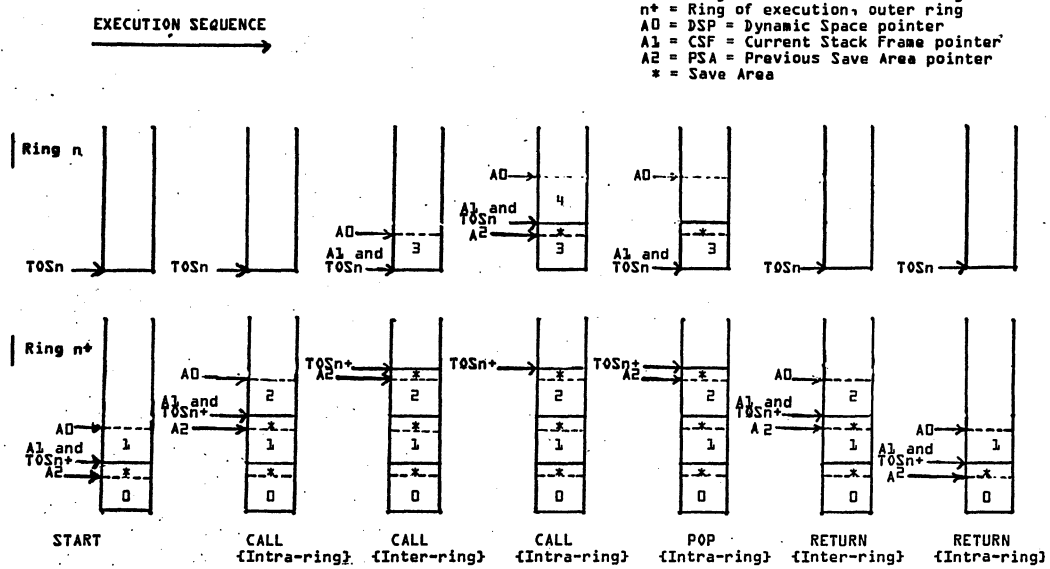


Figure 2.6-1: Call/Return/Pop, post-execution stack frame states (including the Software updating of the Dynamic Space Pointer)

NCR/CDC PRIVATE

2.6.1.6 Exchange
120jk

When executed in Monitor mode this instruction shall change the processor from monitor process state to job process state. See 2.8.5.2.

When executed in Job mode this instruction shall change the processor from job process state to monitor process state. See 2.8.5.1. In addition, the System Call bit in position 10 of the Monitor Condition Register, job process state, shall be set.

The PVA contained in Word 0 (P Register) of the Exchange Package associated with the state from which the exchange is taking place, shall be updated such that it points to the instruction which would have been executed had the exchange not taken place, i.e., the PVA of the "Exchange" instruction with 2 added to its BN field.

The j and k fields from this instruction shall not be translated and their values shall have no effect on the execution of this instruction.

2.6.1.7 Keypoint, class j, code equal to (Xk) Right plus Q
136 jkQ

Operation - The Keypoint instruction shall cause an interrupt to occur if the keypoint mask bit is set which corresponds to the keypoint class number in the j field of the keypoint instruction. Before the interrupt occurs the 4 bits of the j field shall be copied into the keypoint class number (KCN) register and the 32-bit keypoint code shall be copied into the keypoint code register. See 2.5.2.4, 2.5.2.11, and 2.5.2.12.

The keypoint code shall be formed by the addition of Q, expanded to 32 bits by means of sign extension, to the contents of Register Xk Right. For the purpose of this instruction, the contents of Register X0 Right shall be interpreted as consisting of zeros. Arithmetic Overflow shall not be detected during the formation of the keypoint code.

The Keypoint Interrupt shall be recorded in bit 06 of the User Condition Register as described in 2.8.3.7.

NCR/CDC PRIVATE

2.6.1.8 Compare {Xk} at {Aj}; if not equal, load Xk from {Aj}; if equal, store {X0} at {Aj}
125 jk

This instruction shall compare the 64-bit word in Register Xk with the 64-bit word in central memory whose PVA is contained in Register Aj. If equality is found, the contents of Register X0 shall be stored in central memory at the PVA contained in Register Aj. If equality is not found, Register Xk shall be loaded with the central memory word whose PVA is contained in Register Aj.

A serialization function shall be performed before this instruction begins and again at its ending. Execution of this instruction shall be delayed until all previous accesses to central memory on the part of this processor are completed. Execution of subsequent instructions shall be delayed until all central memory accesses due to this instruction are completed.

Conceptually, the execution of this "Compare" instruction on the part of a processor shall result in preventing other processors from accessing the central memory word at the PVA contained in Register Aj between the read and write accesses associated with the execution of this instruction, provided such processors are also executing a "Compare" instruction. With respect to this instruction only, in order to satisfy its "non-preemptive" requirement, the use of 64-bit words consisting entirely of ones in their leftmost 32-bit positions, 00 through 31, shall be reserved for each processor's implementation of this instruction. When the 32-bit halfword initially contained in Register X0 Left consists entirely of ones, an Instruction Specification Error shall be detected, the execution of this instruction shall be inhibited, and the corresponding program interruption shall occur.

Notes: For the purpose of establishing operand access validation, the central memory operand access types shall consist of both a read and a write access. Moreover, those processors having a Cache shall bypass it with respect to the read access and shall purge the associated entry from it with respect to the write access, {See 2.9}. Unless the central memory operand address consists of a byte address which is 0, modulo 8, an Address Specification error shall be detected, the execution of this instruction shall be inhibited, and the corresponding interruption shall occur.

NCR/CDC PRIVATE

2.6.1.9 Load Bit to Xk Right from {Aj} bit indexed by {X0} Right and set bit in central memory
124 jk

Operation - This instruction shall transfer a single bit into Register Xk Right, bit position 63, from a bit position in central memory. This instruction shall also clear the Xk Register in its leftmost 63 bit positions, 00 through 62. The bit position in central memory shall be unconditionally set without changing any other bit positions within the byte or word.

No other processor shall be permitted access to the byte in central memory from the beginning of the read access until the end of the write access which sets the bit within that byte.

A serialization function shall be performed before this instruction begins and again at its ending. Execution of this instruction shall be delayed until all previous accesses to central memory by this processor are completed. Fetching or execution of subsequent instructions by this processor shall be delayed until all central memory accesses from this instruction are completed.

Addressing - The byte in central memory, containing the bit position to be loaded shall be addressed by means of the PVA contained in the Aj Register modified by a bit item count, consisting of a 32-bit index, as follows: The 32-bit halfword obtained from Register X0 Right shall be shifted right three bit positions, end-off with sign extension on the left, and the 32-bit shifted result shall be added to the rightmost 32 bits of the PVA obtained from the Aj Register.

Bit Select - The bit position within the addressed byte in central memory shall be selected by means of the rightmost three bits obtained from Register X0 Right, bit positions 61 through 63. Values from 0 through 7 for these three bits shall select the corresponding bit position, 0 through 7 from the central memory byte.

Notes: For the purpose of establishing access validity, the central memory operand access types shall consist of a read and a write access. Moreover, those processors having a Cache {See 2.9} shall bypass it with respect to the read access and shall purge the associated entry from it with respect to the write access.

NCR/CDC PRIVATE

2.6.1.10 Test Page {Aj} and Set Xk Right
126 jk

This instruction shall test for the presence of the page in central memory corresponding to the PVA contained in Register Aj.

When this instruction finds the corresponding page in central memory, the "Used" bit in the UM field of the associated Page Descriptor shall be set, {See 3.5.1.1}, and the Real Memory Address {RMA} translated from the PVA contained in Register Aj shall be transferred to Register Xk Right.

When this instruction cannot find the corresponding page in central memory, Register Xk Right shall be set in all 32 bit positions, 32 through 63.

Note: Central memory Access Validation {per Section 3.6} shall not be performed during the execution of this instruction

2.6.1.11 Interrupt Product to Xk Right
123 jk

This instruction shall transfer the bit-by-bit product of the Monitor Mask and Monitor Condition Register to bit positions 32 through 47 of Register Xk Right and shall transfer the bit-by-bit product of the User Mask and User Condition Register to bit positions 48 through 63 of Register Xk Right. {See tables 2.8-1 and 2.8-2 for the bit definitions of the Mask and Condition Registers}.

2.6.1.12 Copy to Xk from Central Memory Maintenance Register at {Xj} Right
132jk

This instruction shall copy the central memory Maintenance Register specified by the contents of Register Xj into the Xk Register. All 64 bits of the Xk Register shall be cleared before the selected register is copied into it.

The Processor Memory Port to be utilized during the execution of this instruction shall be determined in the manner defined in subparagraph 2.10.1.1 of this specification with the exception that, for this instruction, bit 33 of the Xj Register shall be used in place of bit 01 of the Real Memory Address as described in that subparagraph, item a.

Bits 56 through 63 of the Xj Register shall contain the number of the central memory Maintenance Register, the contents of which shall be copied into the Xk Register. See Section 4.5 of this specification for central memory Maintenance Register definitions.

NCR/CDC PRIVATE

2.6.2 Local Privileged Mode

This class of instructions shall be permitted to execute only when the processor is in either local privileged mode or in global privileged mode. If an instruction in the local privileged mode class, attempts execution when the processor is not in either local privileged mode or in global privileged mode, a Privileged Instruction Fault shall be detected, execution of that instruction shall be inhibited, and the corresponding program interruption shall occur.

Instructions in the local privileged mode class shall be executable whenever a processor is executing instructions from a segment whose Segment Descriptor defines that segment as either a local privileged executable segment or a global privileged executable segment. See 3.3.1.1}

2.6.2.1 Copy to Central Memory Maintenance Register at Xj Right from Xk
133 jk

This local privileged instruction shall copy the contents of Register Xk into the Central Memory Maintenance Register selected by the contents of Register Xj.

The Processor Memory Port to be utilized during the execution of this instruction shall be determined in the manner defined in subparagraph 2.10.1.1 of this specification with the exception that, for this instruction, bit 33 of the Xj Register shall be used in place of bit 01 of the Real Memory Address as described in that subparagraph, item a.
Bits 56 through 63 of the Xj Register shall contain the number of the central memory Maintenance Register into which the contents of Register Xk shall be copied. See Section 4.5 of this Specification for central memory Maintenance Register definitions.

NCR/CDC PRIVATE

2.6.2.2 Load Page Table Index per {Xj} to Xk and Set X1 Right
127jk

This local privileged instruction shall search the Page Table in central memory, shall return the final index value to Register Xk, and shall set Register X1 Right according to the results of the search.

The entry searched for within the Page Table shall be defined by the System Virtual Address {SVA} contained in Register Xj. For a description of the format for an SVA in an X Register, see subparagraph 2.6.5.3 of this specification.

The Page Table shall be searched in the manner normally employed by the Virtual Addressing Mechanism. Thus, the SVA shall be psuedo-randomized {hashed}, in conjunction with the Page Table Length {PTL}, in order to obtain a nominal index value in the manner described in subparagraph 3.5.2.1 of this specification. The Page Table Address {PTA} interpreted as equal to 0, modulo the PTL, shall be concatenated to this nominal index value for the purpose of determining the first location to be searched in the Page Table.

Beginning with this location, the Page Table shall be linearly searched, {with the nominal index value increased by 8 for each entry which does not correspond to the SVA but does contain a Continue bit equal to 1, up to a maximum of 32 entries searched} in the manner described in subparagraph 3.5.2.2 of this specification.

When a Page Descriptor corresponding to the SVA initially contained in Register Xj is found, the index into the Page Table which is associated with that entry shall be transferred to Register Xk and Register X1 Right shall be set in all 32-bit positions, 32 through 63.

When the Page Table search terminates as a result of not finding a Page Descriptor which corresponds to the SVA initially contained in Register Xj, {whether the termination results from a Continue bit equal to 0 or performing a maximum of 32 comparisons}, the index into the Page Table associated with the last entry compared shall be transferred into Register Xj and the count of entries searched shall be transferred to Register X1 Right.

2.6.3 Global Privileged Mode

This class of instructions shall be permitted to execute only when the processor is in global privileged mode. If an instruction in the global privileged mode class attempts execution when the processor is not in global privileged mode a Privileged Instruction Fault shall be detected, execution of that instruction shall be inhibited, and the corresponding program interruption shall occur.

Global privileged mode shall exist whenever the processor is executing instructions from a segment whose Segment Descriptor defines that segment as a global privileged executable segment. See 3.3.1.1

2.6.3.1 Interrupt Processor per {Xk}.

122 jk

The execution of this global privileged class instruction shall send an external interrupt to one or more other processors via their central memory ports. The processors shall be identified by the central memory port number to which they are connected.

The interrupting processor shall send the contents of Register Xk Right to central memory. Central memory shall then send an external interrupt to the processor{s} on those ports whose port numbers, plus 32, correspond to the bit positions, which are set within Register Xk Right. When the interrupting processor has two ports connected to the same memory {See Figure 1.3-1}, a "switch" shall select the port used to transmit the contents of Register Xk Right to central memory along with the "interrupt" function. {See 2.10.1.1 for "switch" definition}

When the interrupting processor has two ports connected to independent memories {See Figure 1.3-3} the state of Bit 00 of Register Xk shall select the port used to transmit the contents of Register Xk Right to central memory along with the "interrupt" function. When Bit 00 is clear, Port 0 shall be used; when Bit 00 is set, Port 1 shall be used.

j = ?

A serialization function shall be performed before this instruction begins execution. That is, execution of this instruction shall be delayed until all previous central memory accesses on the part of the interrupting processor are complete.

2.6.4 Monitor Mode

This class of instructions shall be permitted to execute only when the processor is in monitor mode. If an instruction in the monitor mode class attempts execution when the processor is not in monitor mode, an Instruction Specification error shall be detected. Execution of that instruction shall be inhibited, and the corresponding program interruption shall occur.

Monitor mode shall exist whenever the processor is in the state defined by the Monitor Exchange Package. The address contained in the Monitor Process State Register shall point to the Monitor Exchange Package.

Note: No single operation code shall be confined to Monitor mode execution. However, sub-operation codes for the instructions defined in 2.6.5 are confined to Monitor mode according to the descriptions contained within that paragraph of this specification.

NCR/CDC PRIVATE

2.6.5 Mixed Mode

This class of instructions shall include those instructions whose mode is dependent on a parameter selection within the instruction. Depending on the value of the parameter, the mode of the instruction shall be non-privileged, local privileged, global privileged, or monitor. The description of each instruction shall define which parameter selects the mode and how the selection is made.

2.6.5.1 Branch to {P} displaced by 2*Q and alter Condition Register per jk.

134 jkQ

This instruction shall test the value of a selected bit in the Condition Register. The j field selects the bit number within the Monitor Condition Register or within the User Condition Register depending on the k field. The k field shall also determine the branch decision and Condition Register bit alteration as follows:

k = 0 or 8, if bit j of the Monitor Condition Register is set, clear it and take a branch exit.

k = 1 or 9, if bit j of the Monitor Condition Register is not set, set it and take a branch exit.

k = 2 or A, if bit j of the Monitor Condition Register is set, take a branch exit.

k = 3 or B, if bit j of the Monitor Condition Register is not set, take a branch exit.

k = 4 or C, if bit j of the User Condition Register is set, clear it and take a branch exit.

k = 5 or D, if bit j of the User Condition Register is not set, set it and take a branch exit.

k = 6 or E, if bit j of the User Condition Register is set, take a branch exit.

k = 7 or F, if bit j of the User Condition Register is not set, take a branch exit.

NCR/CDC PRIVATE

Normal exit - When the test of bit j does not satisfy the branch condition as specified by the k field of this instruction, a normal exit from this instruction shall be performed. A normal exit from this 32-bit instruction shall consist of adding 4 to the rightmost 32 bits of the PVA contained in the P Register, with the sum returned to the P Register's rightmost 32 bits.

Branch Exit - When the test of bit j satisifes the branch condition as specified by the k field of this instruction, a branch exit from this instruction shall be performed. A branch exit shall consist of expanding the @ field from the instruction to 34 bits by means of sign extension, shifting these 34 bits left one bit position with a zero inserted on the right, and adding the 32 bit result to the rightmost 32 bits of the PVA contained in the P Register with the sum returned to the P Register's rightmost 32 bits.

Monitor and Privileged Mode - Some values of the k field of this instruction shall cause this instruction to be a Monitor or Non-privileged instruction as follows:

k	Mode
0 or 8	Monitor
1 or 9	Monitor
2 or A	Non-privileged
3 or B	Non-privileged
4 or C	Non-privileged
5 or D	Non-privileged
6 or E	Non-privileged
7 or F	Non-privileged

Unless the processor is in monitor mode when execution is restricted to monitor mode, an Instruction Specification Error shall be detected, execution of this instruction shall be inhibited, and the corresponding program interruption shall occur.

2.6.5.2 Copy

This group of instructions shall provide the means to copy an X Register to/from a state register. The state register shall be addressed by the contents of the rightmost 8 bits [bits 56 through 63] of Register Xj Right. The list which follows shall define the address assigned to each state register and the bit number location of the register within a word. State registers with addresses in the range A0 through BF shall be read-only with respect to the processor, although the Service Processor may read and write them. State Registers with addresses in the range C0 through FF shall be read or written from the Service Processor only.

Some implementations of this GDS may not use separate flip flop registers. Some state registers may be held in central memory even when they are in active use. For such cases these copy instructions shall make state registers held in central memory appear to operate as copy instructions and not as load or store instructions.

Unless the processor is in Monitor mode when execution is restricted to Monitor mode, an Instruction Specification error shall be detected, execution of this instruction shall be inhibited, and the corresponding program interruption shall occur.

Unless the processor is in the appropriately privileged-mode when execution is restricted to local or global privileged mode, a Privileged Instruction error shall be detected, the execution of this instruction shall be inhibited, and the corresponding program interruption shall occur.

See Table 2.6-1 for register definitions.

Note: Multiple Address assignments have been specified for certain Registers so that, by properly choosing the appropriate address, the contents of a single X Register may be used as both the address and data value for the purpose of copying into such Registers.

Register Number	Register Name	Position in an X Reg.	Position in an Exchange Pkg (word/Bits	Write Mode
00-01	Critical Frame Flag	63	2/00	Unprivileged
02-03	On Condition Flag	63	2/01	↓
04	Debug Index	56-63	36/00-07	
05	Debug Mask Register	59-63	36/11-15	
06	User Mask Register Register	48-63	3/00-15*	
20-23	Trap Enables	62-63	2/14-15	Local
24	Trap Pointer	16-63	35/16-63	Privileged
25	Debug Pointer	16-63	36/16-63	↓
26	Keypoint Mask	48-63	1/00-15	
27	Keypoint Code	32-63	9,10/00-15	
28	Keypoint Class Number	60-63	7/04-07	
29	Process Interval Timer	32-63	11,12/00-15	↓
40	Processor Test Mode	**	Processor	Global Privileged
60	Processor Fault Status	**	Processor	↓
80	Monitor Mask Register	48-63	4/00-15	Monitor
81	Job Process State Pointer	32-63	Processor	↓
82	Page Table Address	32-63	↓	
83	Page Table Length	56-63		
84	Page Size Mask	57-63		
85	System Interval Timer	32-63		
86	One Megahertz Counter	00-63	↓	↓
A0	Processor Identification	40-63	Processor	Non-Writable
A1	Monitor Process State Pointer	32-63	↓	by
A2	Monitor Condition Register	48-63	6/00-15	processor
A3	User Condition Register	48-63	5/00-15	"Copy"
A4	Untranslatable Pointer	16-63	34/16-63	↓
A5	Model Dependent Flags	48-63	15/00-15	
A6	Model Dependent Word	00-63	33/00-63	
A7	Segment Table Length	52-63	16/04-15	
A8	Segment Table Address	32-63	34,35/00-15	
A9	Base Constant	32-63	13,14/00-15	↓
C0	P Register	00-63	0/00-63	Service
C1	Environment Control	**	Processor	Processor
			↓	only

* Bits 0 through 06 are permanently set.
** Processor model dependent.

Table 2.6-1 Register Definitions for "Copy" instructions.

NCR/CDC PRIVATE

a. Copy to Xk per (Xj)
130jk

This instruction shall copy the contents of the state register addressed by the contents of Register Xj into Register Xk. The address assignments and bit locations shall be those defined in Table 2.6-1 The Xk Register shall be cleared before the state register is copied into it.

This instruction shall be a non-privileged instruction.

b. Copy from Xk per (Xj)
131jk

This instruction shall copy the contents of Register Xk into the state register addressed by the contents of Register Xj. The address assignments and bit locations shall be defined in Table 2.6-1 Several of the registers can not be written into by this instruction, as previously described in this sub-paragraph.

Monitor and Privileged Mode - This instruction shall have its mode determined by the state register address contained in Register Xj as follows:

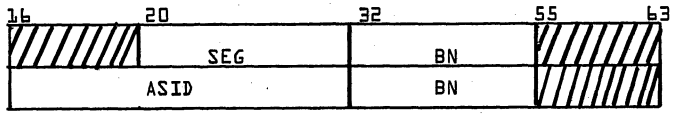
Mode	State Register Number
Unprivileged	00 through 1F
Local privileged, Test Mode	20 through 3F
Global privileged, Test Mode	40 through 7F
Monitor	80 through 9F
Non-writable by processor "Copy"	A0 through BF
Service Processor only	C0 through FF

NCR/CDC PRIVATE

2.6.5.3. Purge Buffer k of Entry per {Xj}
138 jk

Operation - The Purge Buffer instruction shall invalidate entries in the Map and Cache buffers. The purge may invalidate all entries in a buffer, invalidate all entries in a buffer which derive from a given segment, invalidate all entries in a buffer for a given page, or invalidate all entries in a buffer for a given 512 byte block. Register Xj shall contain the required address information, either System Virtual Address {SVA} or Process Virtual Address {PVA}.

An SVA shall contain the Active Segment {ASID} in bits 16 through 31 of Register Xj. A PVA shall contain the Segment number {SEG} in bits 20 through 31 of Register Xj. Bits 32 through 63 shall contain the Byte Number {BN} for either an SVA or a PVA. The rightmost 9 bits of the BN shall be ignored and assumed to be zeros since the smallest purgeable portion of a buffer shall be a 512 byte page or a 512 byte block of a larger page. Proportionately more rightmost bits of the BN shall be ignored and assumed to be zero as page size becomes larger than the 512 byte minimum.



The value of k shall determine the buffer to be purged, the range of entries to be purged, and the type of addressing used to determine the range of entries to be purged. The definition of k follows.

k=0, Purge all entries in Cache which are included in the 512 byte block defined by the SVA in Xj.
k=1, Purge all entries in Cache which are included in the ASID defined by the SVA in Xj.

NCR/CDC PRIVATE

k=2, Purge all entries in Cache.

k=3, Purge all entries in Cache which are included in the 512 byte block defined by the PVA in Xj.
k=4→7, Purge all entries in Cache which are included in the SEG defined by the PVA in Xj.

k=8, Purge all entries in Map which are included in the page defined by the SVA in Xj. The size of the page involved shall be determined by the contents of the Page Size Mask Register.
k=9, Purge all entries in Map which are included in the ASID defined by the SVA in Xj.

k=A→F, Purge all entries in Map.

For k = 0, 1, 9, 8 → F this instruction shall be a local privileged instruction. It shall be non-privileged for all other values of k.

A serialization function shall be performed before this instruction begins execution and again when it completes execution. Execution of this instruction shall be delayed until all previous accesses to central memory, on the part of this processor, are completed. Fetching or execution of subsequent instructions shall be delayed until all central memory accesses due to this instruction are completed.

The implementation of this instruction shall be processor model dependent in that some processor models may not have a Map and/or Cache buffer and they may invalidate more than the required buffer entries. A processor which does not have a Cache shall execute this instruction as a no operation instruction when cache purges are called for by this instruction. Likewise, a processor which does not have a Map shall execute this instruction as a no operation instruction when map purges are called for by this instruction. The processor model dependent specifications shall fully define these model dependent charactistics.

NCR/CDC PRIVATE

2.7 Program Monitoring

2.7.1 Performance Monitoring

Performance of the overall software/hardware system shall be monitored via the insertion of Keypoint instructions at "key" points in the software. Each Keypoint instruction shall be identified by its class and by its code within each class. Keypoint classes and keypoint codes shall be assigned such that system performance data shall be determined from the order and frequency of the occurrence of keypoint instructions of various classes and codes.

Two methods of gathering the keypoint data shall be provided. The first method shall be via software internal to the processor. The second method shall be via an external hardware device. See paragraph 2.10.3 of this specification.

2.7.1.1 Keypoint Software Recording

Software Recording of keypoint data shall be based on the occurrence of a trap whenever the keypoint class number in the keypoint instruction matches a mask bit in the keypoint trap mask register. The trap routine shall record the keypoint class number, keypoint code and the time. Another routine shall subsequently analyze the data gathered by the keypoint trap routine to produce the system performance data.

2.7.1.2 Keypoint Hardware Recording

Hardware recording of keypoint data may be accomplished via an external hardware device which shall record the keypoint events as they occur. Each time a keypoint instruction is executed the hardware keypoint recorder may record the keypoint class number, keypoint code and time.

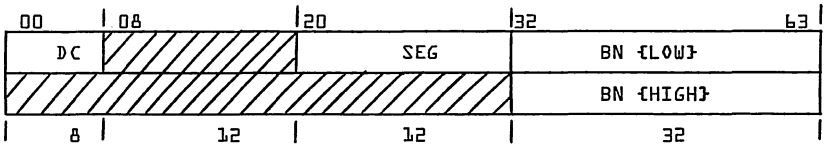
Hardware keypoint recording gives minimum interference to program execution since the keypoint event data can be recorded without a trap taking place. The CPU shall provide a signal to the keypoint monitor device indicating that a keypoint instruction is being performed. The CPU shall also supply the keypoint monitor device with the keypoint class and keypoint code for the keypoint instruction being performed.

NCR/CDC PRIVATE

2.7.2 Debug

2.7.2.1 Debug List

The format of a Debug List entry is:



where DC is the Debug Code, BN (LOW) is the byte number of the beginning of the contiguous field in memory to which the Debug Code applies, BN (HIGH) is the byte number of the last byte in the contiguous field in memory to which the Debug Code applies, and SEG is the process segment number to which the Debug Code applies.

Debug List entries shall be aligned on word boundaries.

The Debug List shall not be longer than 32 entries (64 words) in length. Entries beyond 32 shall be ignored.

The matching of BN (LOW) and BN (HIGH) shall be against the address of the leftmost byte of a piece of information only; whether it is a word, halfword, byte string, or 16-bit instruction. The matching shall include the end points. That is:

$$BN\ (LOW) \leq Address \leq BN\ (HIGH).$$

The first entry in the Debug List shall be at the PVA contained in the Debug List Pointer Register (see 2.5.2.23). Position within the Debug List during a scan shall be kept track of by the Debug List Index Register (see 2.5.2.22).

2.7.2.2 Debug Code (DC)

The DC bit assignments are:

Bit 0: Data Read, first address of string - applies to all central memory accesses that are defined as read accesses for purposes of access protection.

NCR/CDC PRIVATE

Bit 1: Data Write, first address of string-applies to all central memory accesses that are defined as write accesses in the memory protection system.

Bit 2: Instruction Fetch
Applies to all central memory accesses that are defined as an execution access in the memory protection system. Note that the instruction fetch from memory will have already occurred.

Bit 3: Branching instruction

Applies to branch and return instructions which, when executed will result in a branch exit to the next instruction. The address bracket shall apply to the address of the instruction branched to.

Bit 4: Call instruction

Applies to the occurrence of either Call instruction (instruction reference number 115 or 116). The address bracket shall apply to the address of the called procedure.

Bit 5: End of list

Denotes that this is the last entry in the Debug List

More than one bit may be set in a DC entry. The End of List DC (bit 5) shall be interpreted after all other bits in the same DC have been interpreted and acted upon.

2.7.2.3 Debug Operation

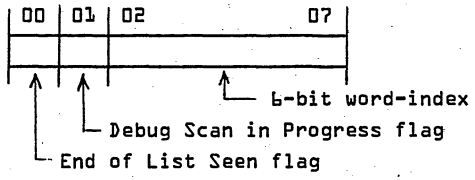
The Debug List shall be scanned after instruction fetch but prior to instruction execution, provided traps are enabled, the Debug Mask bit in the User Mask Register is set, one or more bits in the Debug Mask Register are applicable to the instruction to be executed, and the End of List Seen Flag in the Debug Index Register is clear.

When the Debug Scan in Progress flag is clear, the Debug List shall be scanned by reading the first word from the Debug List in central memory at the PVA specified by the contents of the Debug List Pointer Register. After the first word of the Debug List has been read, the Debug Scan in Progress flag in the Debug Index Register shall be set and each successive word from the Debug List shall be read by incrementing the 6-bit word-index

NCR/CDC PRIVATE

field contained in the Debug Index Register by one, and referencing the Debug List at the PVA specified by the initial contents of the Debug List Pointer Register, modified in its rightmost 32-bit positions by the addition of the zero-extended, rightmost 6-bits of the Debug Index Register.

The Debug Index Register, contained in bit positions 00 through 07 of word 36 in the Exchange Package, shall be formatted as follows:



When one or more bits contained in the Debug Mask Register are set and are equal to one or more of the corresponding leftmost 5 bits of the Debug Code contained in the first word of a doubleword entry in the Debug List, and one or more of the appropriate PVAs associated with the instruction's execution are contained within the address bracket defined by the corresponding double word entry from the Debug List, the Debug bit in the User Condition Register shall be set, the execution of the instruction shall be inhibited and a trap interrupt shall occur. Moreover, when the End of List bit in Debug Code is set or the 32nd double word entry from Debug List has been scanned, the End of List Seen Flag contained in the Debug Index Register shall be set.

The second word of the double word Debug List entry which causes a trap interrupt shall be identifiable by the value of the rightmost 6-bits of the Debug Index Register and the PVA contained in the Debug List Pointer Register.

The Debug Index Register shall also provide the means for properly initiating, resuming and terminating Debug Scan operations, (particularly when an instruction's execution has been inhibited by one or more program interrupts occurring on the one or more immediately preceding fetches of the associated instruction). Thus, the clearing of the Debug Index Register shall occur, conceptually, at the completion of each instruction's execution for which a Debug Scan has immediately preceded that execution.

For the purpose of establishing central memory access validation, each central memory access performed for the purpose of reading a word from the Debug List as a part of a Debug Scan operation, shall be a read type access.

NCR/CDC PRIVATE

2.8 Interrupts

There shall be two classes of interrupts. One class shall be called an Exchange Interrupt. The other class shall be called a Trap Interrupt.

Exchange interrupts shall normally originate from bits in the Monitor Condition Register and shall cause an exchange operation from Job Mode to Monitor Mode to occur. The Monitor Mask Register shall enable selected conditions in the Monitor Condition Register to generate an exchange interrupt.

Trap interrupts shall normally originate from bits in the User Condition Register and shall cause a trap operation to occur. The User Mask Register shall enable selected conditions in the User Condition Register to generate a trap interrupt.

2.8.1 Monitor Condition Register

The Monitor Condition Register shall contain 16 bits. Each bit, when set, shall indicate that a particular condition has occurred in the processor or that the processor has been informed of an event which occurred external to itself.

When a bit in the Monitor Condition Register is set and the processor is in Job Mode, an Exchange interrupt shall occur if the associated bit in the Monitor Mask Register is set.

When a bit in the Monitor Condition Register is set and the processor is in Monitor Mode, a Trap Interrupt shall occur if the associated bit in the Monitor Mask Register is set, the Trap Enable Flip-Flop is set, and Trap Enable Delay is clear.

Table 2.8-1 shall define the bit number assignments for the Monitor Condition Register and the action to be taken for each bit under various circumstances.

A Monitor Condition Register bit once set, shall remain set until cleared by a master clear, by a "Branch and alter Condition Register" instruction (reference number 134), provided the instruction has specified that the bit shall be cleared, or by alteration when the Condition Register is held in an Exchange Package in central memory.

Each bit in the Monitor Condition Register is defined in the sections which follow.

NCR/CDC PRIVATE

Monitor Condition or Mask Bit Number	Monitor Condition Name	Action taken when Condition occurs: Job Mode, Mask Bit Set	Action taken when Condition occurs: Monitor Mode, Trap Enabled *	Action taken when Condition occurs: Monitor Mode, Trap Disabled
00	Processor Malfunction	Exchange	Trap	Halt
01	Memory Malfunction	Exchange	Trap	Halt
02	Real Address Out of Range	Exchange	Trap	Halt
03	Instruction Specification Error	Exchange	Trap	Halt
04	Address Specification Error	Exchange	Trap	Halt
05	Invalid Segment	Exchange	Trap	Halt
06	Access Violation	Exchange	Trap	Halt
07	Environment Specification Error	Exchange	Trap	Halt
08	External Interrupt	Exchange	Trap	Stack
09	Page Table Search Without Find	Exchange	Trap	Halt
10	System Call	Exchange	Trap	Stack
11	System Interval Timer	Exchange	Trap	Stack
12	Ring Number Zero	Exchange	Trap	Halt
13	Outward Call/Inward Return	Exchange	Trap	Halt
14	Software Error Log	Exchange	Trap	Stack
15	Trap Exception	Exchange	Trap	Halt

Halt - Stop processor. SP can intervene and restart processor.
Stack - Test for opportunity to Trap or Exchange, each instruction fetch
Job Mode, Mask Bit Clear - Same as Monitor Mode, Trap Disabled
* Trap Enabled - See 2.8.1 for the definition of "Trap Enabled"

Table 2.8-1

Monitor Condition Register and Mask Register Bit Definitions

NCR/CDC PRIVATE

2.8.1.1 Processor Malfunction

The processor malfunction bit in the Monitor Condition Register, if set, shall indicate that a condition has been detected in the processor which is not indicated by other Condition Register bits. These conditions shall include the following:

a. Power failure imminent
b. Excessive temperature
c. Other processor model dependent conditions as specified in the processor model dependent specification.

The PVA which shall be stored when a Processor Malfunction interrupt occurs shall point to the instruction whose execution was not completed because of the Processor Malfunction.

2.8.1.2 Memory Malfunction

The memory malfunction bit in the Monitor Condition Register, if set, shall indicate that a condition has been detected in a central memory unit which is used by this processor but is not indicated by other Condition Register bits. These conditions shall include the following conditions.

a. Address, data transmission, or function parity errors on central memory accesses from this processor.
b. Non-correctable central memory data parity errors on central memory accesses from this processor.
c. Power failure imminent in central memory
d. Excessive temperature in central memory

The PVA which shall be stored when a Memory Malfunction interrupt occurs shall point to the instruction whose execution was not completed due to the Memory Malfunction.

2.8.1.3 Real Address Out of Range

The real address out of range bit in the Monitor Condition Register, if set, shall indicate that the real address used for central memory access by this processor is for a central memory address which is not physically accessable by the processor. See 3.1.3.

The PVA which shall be stored when a Real Address Out of Range interrupt occurs shall point to the instruction which made the

NCR/CDC PRIVATE

central memory access to the address which is out of range.

2.8.1.4 Instruction Specification Error

The instruction specification error bit in the Monitor Condition Register, if set, shall indicate that one of the following errors has occurred.

a. Length Specification errors as described in paragraph 2.2.9 and subparagraphs 2.2.1.3, 2.3. 2.3, and 2.3.4.5 and 2.2.10.
b. Type Specification errors as described in paragraph 2.3.3 as well as all type combinations, other than those defined as valid, for the instructions described in each subparagraph of paragraphs 2.3.4 through 2.3.7.
c. Field Descriptor errors resulting from combinations, other than those defined as valid, for each of the instructions described in subparagraphs 2.3.5.1 through 2.3.5.4.
d. Instruction Specification errors as described for the Calculate Subscript instruction in subparagraph 2.3.5.5.
e. Execution of a Program Error instruction as described in subparagraph 2.6.1.1.
f. Execution of a Monitor Mode operation when the processor is not in Monitor mode. See 2.6. 4 and 2.6.5.

The PVA which shall be stored when an Instruction Specification Error interrupt occurs shall point to the instruction with the faulty specification.

2.8.1.5 Address Specification Error

The address specification error bit in the Monitor Condition Register, if set, shall indicate that an attempt was made to use an improper address. Improper addresses shall include:

a. The address modulus defined for specified instructions or specified registers is not met. See 2.6.1.2 through 2.6.1.5.
b. Other address bit(s) defined as zero(s) for specified instructions or specified registers are not zero(s). See 3.2.1.3.

The PVA which shall be stored shall point to the instruction which made the central memory access that caused the Address Specification interrupt to occur.

2.8.1.6 Invalid Segment

The invalid segment bit in the Monitor Condition Register, if set,

NCR/CDC PRIVATE

shall indicate that a PVA could not be translated into a real memory address for one of the following reasons.

a. Segment table length exceeded. See 3.3.
b. Invalid segment descriptor element. See 3.3.1.1.

The PVA which shall be stored for an Invalid Segment interrupt shall point to the instruction which attempted the central memory access which resulted in the Invalid Segment Condition.

2.8.1.7 Access Violation

The access violation bit in the Monitor Condition Register, if set, shall indicate that the requested memory access was blocked because it did not have the required access permission. Access violations shall be detected for the following central memory access situations. See section 3.6 of this specification for details.

a. Read central memory when read access is not granted or read is not within read ring limits.
b. Write central memory when write access is not granted or write is not within write ring limits.
c. Attempt to execute when execute access is not granted or execute is not within execute ring limits.
d. Call via a code base pointer which is not in a binding section segment. See 2.6.1.2.
e. Call from a process beyond the call ring limit. See 2.6.1.2.
f. Key/lock violations. See section 3.6.3.2 for the definition of key/lock violations.

The PVA which shall be stored when an Access Violation interrupt occurs shall point to the instruction which made the central memory access which attempted to violate the access protection mechanism.

2.8.1.8 Environment Specification Error

The environment specification error bit in the Condition Register, if set, shall indicate that an error was detected in the environment during a trap operation, a Return instruction, or a Pop instruction. Environmental specification errors shall be detected for the following reasons.

a. The Code Base Pointer's External Procedure bit is not set for a trap operation. See 2.8.6.
b. The new DSP pointer is not equal to the old PSA pointer for a Return or Pop instruction. See 2.6.1.4 and 2.6.1.5.

The PVA which shall be stored when an environment error interrupt occurs shall point to the instruction which caused environment specification error or to the instruction which would have been executed had the trap interrupt, associated with the environment specification error, not occurred.

2.8.1.9 External Interrupt

The external interrupt bit in the Monitor Condition Register, if set, shall indicate the presence of an interrupt from another processor. {The recipient processor may read a message in central memory to determine who the calling processor is and the purpose of the external interrupt.}

The PVA which shall be stored when an External Interrupt occurs shall point to the instruction which would have been executed if the interrupt had not occurred.

2.8.1.10 Page Table Search Without Find

The page table search without find bit in the Monitor Condition Register, if set, shall indicate that the requested page table entry was not found during the linear search of the page table which begins at the "hashed" entry address and ends a maximum of 32 entries later. Thus, the system virtual address could not be mapped into a real memory address. See 3.5.2.

The PVA which shall be stored for a Page Table Search Without Find interrupt shall point to the instruction which attempted the central memory access which resulted in the Page Table Search Without Find condition.

2.8.1.11 System Call

The system call bit in the Monitor Condition Register, if set, shall indicate that a process has executed an Exchange instruction {reference number 120} which caused an exchange interrupt from job process state to monitor process state. This bit shall not be set by an Exchange instruction going from monitor process state to job process state. See 2.6.1.6.

The PVA which shall be stored for a system call interrupt shall point to the instruction which would have been executed if the interrupt had not occurred.

2.8.1.12 System Interval Timer

The system interval timer bit in the Monitor Condition Register, if set, shall indicate that the System Interval Timer has decremented to a count equal to zero. See 2.5.3.3.

The PVA which shall be stored when a System Interval Timer interrupt occurs shall point to the instruction which would have been executed if the interrupt had not occurred.

2.8.1.13 Ring Number Zero

The ring number zero bit in the Monitor Condition Register, if set, shall indicate that the ring number of a PVA is equal to zero. A ring number equal to zero shall mean that the pointer has not been linked.

The PVA stored when a Ring Number Zero interrupt occurs for a Call instruction shall point to the call instruction which attempted the central memory access which resulted in the Ring Number Zero condition. See 2.6.1.2.

The PVA stored when a Ring Number Zero interrupt occurs for a "Load A" instruction shall point to the next instruction which would have been executed if the interrupt had not occurred. See 2.2.1.6 and 2.2.1.7.

2.8.1.14 Outward Call/Inward Return

The outward call/inward return bit in the Monitor Condition Register, if set, shall indicate that an outward call or an inward return has been attempted by the processor. An outward call shall have been attempted if the call instruction attempts a call to a procedure with a ring number larger than the ring number of the procedure which contains the call instruction. An inward return shall have been attempted if the return instruction attempts a return to a procedure with a ring number smaller than the ring number of the procedure which contains the return instruction. See 2.6.1.2 and 2.6.1.4.

The PVA stored when an Outward Call/Inward Return interrupt occurs shall point to the instruction which attempted the outward call or inward return.

2.8.1.15 Software Error Log

The software error log bit in the Monitor Condition Register, if set, shall indicate the need for software to log the occurrence of errors which the hardware has detected and corrected. These error conditions shall include:

a. The central memory corrected error register(s) is full. This error register(s) logs corrected errors in central memory for the port(s) used by this processor.
b. Other processor model dependent errors shall be specified in the appropriate processor model dependent specification.

The PVA which shall be stored when a Software Error Log interrupt occurs shall point to the instruction which would have been executed if the interrupt had not occurred.

2.8.1.16 Trap Exception

The trap exception bit in the Monitor Condition Register, if set, shall indicate that a fault was detected during the trap interrupt operation. The fault detected shall be indicated by setting the appropriate bit in the Monitor Condition Register. Thus at least one other Monitor Condition Register bit shall be set whenever the trap exception bit is set.

The PVA which shall be stored when a trap exception interrupt occurs shall be the PVA of the instruction which would have been executed had the trap interrupt, associated with the trap exception, not occurred.

2.8.2 Monitor Mask Register

The Monitor Mask Register shall be used by the system to enable an exchange interrupt to occur when the processor is in job mode and a bit in the Monitor Condition Register is set. There shall be an individual bit in the Monitor Mask Register for each bit in the Monitor Condition Register.

Table 2.8-1 defines the Monitor Mask Register bit numbers as well as the action that shall take place for each bit in the Monitor Condition Register under various circumstances.

2.8.3 User Condition Register

The User Condition Register shall contain 16 bits. Each bit, when set, shall indicate that a particular condition has occurred

in the processor. When a bit in the User Condition Register sets, a trap interrupt shall occur if the trap interrupt is enabled. Enablement of a trap interrupt shall include - the Trap Enable flip flop is set, the Trap Enable Delay flip flop is not set, and the User Mask Register bit which corresponds to the User Condition Register bit is set.

Table 2.8-2 shall define the bit number assignments for the User Condition Register and the action to be taken by each bit in various environments.

A User Condition Register bit once set, shall remain set until cleared by a master clear, by a "Branch and alter Condition Register" instruction (reference number 134), when the instruction has specified that the bit shall be cleared, or by alteration when the Condition Register is held in an Exchange Package in central memory.

Each bit in the User Condition Register is defined in the sections which follow.

2.8.3.1 Privileged Instruction Fault

The privileged instruction fault bit in the User Condition Register, if set, shall indicate that one of the following faults has occurred.

a. An attempt was made to execute a local privileged instruction in other than local privileged executable mode or in global privileged executable mode. See 2.6.2.
b. An attempt was made to execute a global privileged instruction in other than global privileged executable mode. See 2.6.3.

The PVA which shall be stored shall point to the instruction which caused the Privileged Instruction Fault interrupt to occur.

2.8.3.2 Unimplemented Instruction

The unimplemented instruction bit in the User Condition Register, if set, shall indicate that an instruction operation code which is not implemented in a particular processor model has attempted execution in that processor model. The implementation of this bit is processor model dependent and shall be fully specified in the appropriate processor model dependent specifications.

The PVA which shall be stored shall point to the Unimplemented Instruction which caused the interrupt to occur.

NCR/CDC PRIVATE

2.8.3.3 Free Flag

The free flag bit in the User Condition Register, if set, shall indicate that this process shall take immediate note of a situation which occurred when this process was not in active execution.

A process' free flag shall normally be set in the process' exchange package when the exchange package is in central memory. In this way, a system process shall gain the object process' immediate attention the next time the object process begins active execution.

The PVA which shall be stored shall point to the instruction which would have been executed if the Free Flag interrupt had not occurred.

User Condition or Mask Bit Number	User Condition Name	Action taken when Condition occurs		
			Trap Disabled	
		Trap Enabled **	Job Mode	Monitor Mode
00*	Privileged Instruction Fault	Trap	Exchange	Halt
01*	Unimplemented Instruction	Trap	Exchange	Halt
02*	Free Flag	Trap	Stack	Stack
03*	Process Interval Timer	Trap	Stack	Stack
04*	Inter-ring Pop	Trap	Exchange	Halt
05*	Critical Frame Flag	Trap	Exchange	Halt
06*	Keypoint	Trap	Stack	Stack
07	Divide Fault	Trap	Stack	Stack
08	Debug	Trap	N/A	N/A
09	Arithemtic Overflow	Trap	Stack	Stack
10	Exponent Overflow	Trap	Stack	Stack
11	Exponent Underflow	Trap	Stack	Stack
12	F. P. Loss of Significance	Trap	Stack	Stack
13	F. P. Indefinite	Trap	Stack	Stack
14	Decimal Loss of Significance	Trap	Stack	Stack
15	Invalid BDP Data	Trap	Stack	Stack

*Mask bit is permanently set
Halt - Stop Processor. SP can intervene and restart processor.
Stack - Test for opportunity to Trap or Exchange, each instruction fetch.
N/A - Not Applicable. Debug condition cannot occur when Traps are disabled. (However, once Debug Trap occurs, condition bit should be reset by software prior to re-enabling Trap).
** See 2.8.3 for the definition of "Trap Enabled"

Table 2.8-2
User Condition Register and Mask Register Bit Definitions

NCR/CDC PRIVATE

2.8.3.4 Process Interval Timer

The process interval timer bit in the User Condition Register, if set, shall indicate that the process interval timer has decremented to zero. See 2.5.3.2.

The PVA which shall be stored shall point to the instruction which would have executed if the Process Interval Timer interrupt had not occurred.

2.8.3.5 Inter-ring Pop

The inter-ring pop bit in the User Condition Register, if set, shall indicate that an attempt was made to "pop" a stack frame in one ring with a Pop instruction {reference number 118} executing in a different ring. See 2.6.1.5.

The PVA which shall be stored shall point to the Pop instruction which attempted the inter-ring pop.

2.8.3.6 Critical Frame Flag

The critical frame flag bit in the User Condition Register, if set, shall indicate that an attempt was made to "pop", or "return" from, a critical stack frame. See 2.6.1.4 and 2.6.1.5.

The PVA which shall be stored shall point to the Return instruction or the Pop instruction which attempted to "pop," or "return" from, a critical stack frame.

2.8.3.7 Keypoint

The keypoint bit in the User Condition Register, if set, shall indicate that a selected keypoint instruction has been executed. See section 2.7 and subparagraph 2.6.1.7.

The PVA which shall be stored shall point to the next instruction which would have been executed if the keypoint interrupt had not occurred.

2.8.3.8 Divide Fault

For the definition of this condition, see the instruction descriptions in subparagraphs 2.2.2.4, 2.2.2.9 and 2.3.3.1.

The PVA which shall be stored shall point to the instruction in which the divide fault occurred.

2.8.3.9 Debug

For the definition of this condition, see the debug description in section 2.7.2.

The PVA which shall be stored shall point to the instruction which caused the Debug interrupt to occur. {For the purposes of this definition an instruction fetch shall be considered part of the execution of that instruction and a branch taken shall be considered part of the execution of the branch instruction.}

2.8.3.10 Arithmetic Overflow

For the definition of this condition, see the instruction descriptions in subparagraphs 2.2.2.1 through 2.2.2.9, 2.3.3.3 and 2.4.2.2.

The PVA which shall be stored shall point to the instruction which would have been executed if the Arithmetic Overflow trap interrupt had not occurred.

2.8.3.11 Exponent Overflow

For the definition of this condition, see the descriptions in subparagraph 2.4.1.3 and 2.4.1.6, item a.

The PVA which shall be stored shall point to the instruction which would have been executed if the Exponent Overflow trap interrupt had not occurred.

2.8.3.12 Exponent Underflow

For the definition of this condition, see the descriptions in subparagraphs 2.4.1.3 and 2.4.1.6, item a.

The PVA which shall be stored shall point to the instruction which would have been executed if the Exponent Underflow trap interrupt had not occurred.

2.8.3.13 Floating Point Loss of Significance

For the definition of this condition see subparagraph 2.4.1.6, 2.4.3.1 and 2.4.3.3.

The PVA which shall be stored shall point to the instruction which would have been executed if the Floating Point Loss of Significance trap interrupt had not occurred.

2.8.3.14 Floating Point Indefinite

For the definition of this condition see subparagraphs 2.4.1.3 and 2.4.1.6, item b.

The PVA which shall be stored shall point to the instruction which would have been executed if the Floating Point Indefinite trap interrupt had not occurred.

2.8.3.15 Decimal Loss of Significance

For the definition of this condition see paragraph 2.3.3.

The PVA which shall be stored shall point to the instruction which would have been executed if the Decimal Loss of Significance trap interrupt had not occurred.

2.8.3.16 Invalid BDP Data

For the definition of this condition see paragraph 2.3.3 and subparagraph 2.3.3.3.

The PVA which shall be stored shall point to the instruction which would have been executed if the Invalid BDP Data trap interrupt had not occurred.

2.8.4 User Mask Register

The User Mask Register shall be used by the user processes to permit a trap interrupt to occur when a bit in the User Condition Register is set, provided the corresponding bit in the User Mask Register is set, the Trap Enable Delay Flip-flop is clear, and the Trap Enable Flip-flop is set. There shall be an individual bit in the User Mask Register for each bit in the User Condition

NCR/CDC PRIVATE

Register. A User Mask bit when not set shall disable a trap interrupt from occurring.

Table 2.8-2 defines the User Mask Register bit numbers and also defines the action that shall take place for each bit in the User Condition Register under various circumstances. Bits 00 through 06 of the User Mask Register shall be unconditionally interpreted as equal to ones.

2.8.5 Exchange Interrupt Operation

Exchange interrupts shall be that class of interrupts which cause a change from a Job Process state to the Monitor Process state.

The Monitor Process state shall contain a small program which contains only the most basic operations necessary to recognize and provide for subsequent action on exchange interrupt conditions. The hardware shall disable further exchange interrupts while the processor is in the monitor process state. Exchange interrupts shall be caused by those bits in the Condition Registers which are identified as exchange interrupt conditions in Tables 2.8-1 and 2.8-2.
The exchange package (see Figure 2.5-2) shall be contained in central memory at separate locations for each process. The exchange package shall be used to establish the environment for each process when the process is activated. An exchange operation shall deactivate one process and activate a second process.

The exchange operation shall consist of moving the environment for the current process state into its central memory locations and establishing the environment for the next process state by moving it from its central memory locations.

The number of items in the exchange package held in registers when a state is active shall be processor model dependent and shall be fully specified in the processor model dependent specifications.

The PVA stored in the P Register portion of the Exchange Package, for each condition that can cause an exchange interrupt, is defined in each Condition Register bit definition (see 2.8.1 and 2.8.3). The same definition for the PVA stored shall apply when an exchange interrupt is forced to become a trap interrupt, except that the PVA shall be stored in the Current Stack Frame Save Area instead.

NCR/CDC PRIVATE

2.8.5.1 Job Process to Monitor Process Exchange

The hardware shall perform the following steps when doing a Job Process to Monitor Process exchange.

a. Store the current job process state exchange package in central memory beginning at the address contained in the job process state pointer register.
b. Disable Exchange interrupts.
c. Load the monitor process state exchange package from central memory, beginning at the address contained in the monitor state pointer register, into the processor environment registers.

Exchange interrupt conditions which occur in the monitor process state while traps are enabled shall be trapped.

Exchange interrupt conditions which occur in the monitor process state while traps are disabled shall be held until traps are enabled, in which case, a trap shall be taken; or held until an exchange has been made back to a job process state, in which case an exchange interrupt shall be taken. For those cases in which continued processor execution is impossible or likely to destroy information, the processor shall halt. The SP may then take appropriate action such as terminating the job and/or removing the processor from the system.

See Tables 2.8-1 and 2.8-2 for the definition of how the conditions are handled under various circumstances.

2.8.5.2 Monitor Process to Job Process Exchange

The hardware shall perform the following steps when performing a monitor process to job process exchange, {See 2.6.1.6}.

a. Store the monitor process state exchange package in central memory beginning at the address contained in the monitor process state pointer register.
b. Enable exchange interrupts
c. Load the job process state exchange package into the processor environment registers from central memory beginning at the address contained in the job process state pointer register.

Notes: The monitor process shall establish the next job process for execution by loading the job process state pointer register with the central memory location of the next job's exchange package.

2.8.5.3 External Interrupt

External interrupts are inter-processor interrupts which shall cause an interrupt to occur in the processor which receives such a signal. The purpose of the external interrupt shall be to permit efficient inter-processor communication.

Messages shall be passed between processors via central memory Refer to the IPLOS GDS for details.

External interrupts shall be generated by the "Interrupt Processor per j and {Xk} Right" instruction {Reference number 122}. See 2.6.3.1.

2.8.6 Trap Interrupt Operation

Trap Interrupts shall be accomplished by simulating a Call instruction to an external procedure.

A Trap Frame shall be established in the manner described in subparagraph 2.5.4.1 of this specification. This Trap Frame shall be used to store the "environment" of the "trapped" procedure.

Code Base and Binding Section Pointers shall be obtained by using the PVA contained in the Trap Pointer Register in place of the "{Ai} plus 8*D" as utilized by the explicit Call instruction, {described in subparagraph 2.6.1.2 of this specification}, which the Trap Interrupt shall simulate.

All exception conditions which result from the simulation of the Call instruction, shall result in an Exchange Interrupt. Such exceptional conditions shall set the "Trap Exception" bit in the Condition Register.

The Call instruction shall be simulated by means of the sequence described in 2.6.1.2 with steps i and k omitted and step b accomplished as follows: The Dynamic Space Pointer initially contained in Register A0, shall be increased by 64 {decimal} and the result shall be stored into the process' Exchange Package as the Top of Stack pointer corresponding to the ring of execution of the "trapped" procedure.

Unless the Code Base Pointer's External Procedure bit is equal to a one, an Environment Specification Error shall be detected, the Trap Interrupt shall be inhibitied, and an Exchange Interrupt or a processor halt shall occur. {See Table 2.8-1}.

The trap operation shall be disabled by hardware immediately upon the occurence of a trap. The trap operation may be re-enabled by software after the condition causing the trap has been sensed and cleared. See paragraph 2.5.2.20.

All bits in the Condition Registers which are identified as trap interrupts shall cause a trap interrupt when set, under the circumstances described in Tables 2.8-1 and 2.8-2.

NCR / CDC PRIVATE

User processes shall have control over whether a user condition will cause a trap, via the User Mask Register. Bits in the User Mask Register when set shall permit corresponding User Condition bits to trap. Several of the User Mask Register bits shall be permanently set and are specified in section 2.8.4.

Trap Conditions which occur when traps are not enabled shall in some cases become exchange interrupts. {Also, when the processor is in Monitor mode, the processor shall halt.} Table 2.8-2 defines how each User Condition bit is handled for these circumstances.

The PVA stored in the P Register portion of the Current Stack Frame save area, for each condition that can cause a trap interrupt, is defined in each Condition Register bit definition in paragraphs 2.8.1 and 2.8.3.

2.8.7 Multiple Interrupts

When more than one bit is set in the Condition Registers {Monitor and User} the following priority shall be observed:

a. Exchange interrupts shall be serviced before trap interrupts.

b. Within either type of interrupt, the priority listed in Table 2.8-3 shall be observed.

Priority group 3 in Table 2.8-3 specifies the priority which shall be used in testing for conditions within this group. The first condition found within this group shall cause testing for lower priority conditions to terminate.

NCR / CDC PRIVATE

Priority Group	Condition Register Bit	Group Characteristics
1.	Processor Malfunction Memory Malfunction Real Address Out of Range	Mid-execution faults
2a.	System Interval Timer Software Error Log External Interrupt Free Flag Process Interval Timer	Between Commands, Not Instruction Generated
2b.	System Call Keypoint Arithmetic Overflow Exponent Overflow Exponent Underflow F.P. Loss of Significance F.P. Indefinite Decimal Loss of Significance Invalid BDP Data	Between Commands, Instruction Generated
3.	Instruction Specification Error Address Specification Error Invalid Segment Access Violation Environment Specification Error Page Table Search without Find Ring Number Zero Outward Call/Inward Return Trap Exception Privileged Instruction Fault Unimplemented Instruction Inter-ring Pop Critical Frame Flag Divide Fault Debug	Instruction Generated, Pre-tested before execution.

Table 2.8-3 Condition Registers, Bit Groupings

(Highest priority at top of Group 3, lowest priority at bottom of Group 3, except where otherwise specified by individual instruction descriptions)

NCR/CDC PRIVATE

2.9 Buffers

Two buffers to increase processor performance may be included in the processor. These buffers are described in the following sections. The existence, size, performance, and organization of these buffers shall be processor model dependent.

2.9.1 Map Buffer

The Map Buffer shall be a high speed memory used to eliminate repeated references to the segment tables and the page table.

Map size, operation and entry replacement algorithm shall be processor model dependent.

2.9.2 Cache Buffer

The Cache Buffer shall be a high speed memory which shall be used to reduce the access time to central memory for words which are used more than once.

Cache size, operation, and entry replacement algorithm shall be processor model dependent.

2.10 Interfaces

2.10.1 Central Memory

The processor central memory interface shall be compatible with the central memory interface specified in 4.1.3 & 4.2 of this specification. Compatible shall mean that all signals and operations shall be provided as specified in 4.1.3 & 4.2 except that transmitted signals become received signals and vice versa.

2.10.1.1 Processor Central Memory Port Selection

Each processor shall provide the means for accommodating two ports to central memory as previously described in paragraph 1.3.3 of this specification.

a. When these two ports are connected to independent memories, as illustrated in Figure 1.3-3 of this specification, the processor central memory port used for any given central memory access shall be determined by the state of bit 0_1 of

NCR/CDC PRIVATE

the Real Memory Address, {See 3.1.3}, as used for the central memory access. If the bit is set, port 1 shall be used. If the bit is clear, port 0 shall be used.

Note: When used in this manner, bit 01 of the Real Memory address {RMA} shall not be transmitted to central memory.

b. When only a single port is present, as well as when two ports are connected to the same memory according to the illustration in Figure 1.3-1 of this specification, a "switch" shall select the port to be active at any given point in time. Moreover, the RMA shall be neither translated nor modified in the manner described in item a. of this subparagraph.

Note: At a minimum, the state of this "switch" shall be controlled by the service processor, {SP}.

NCR/CDC PRIVATE

2.10.2 Service Processor

The Service Processor {SP} Interface shall carry the fundamental signals which are necessary for control, maintenance, and initialization of the processor. The SP Interface shall be identical for upper IPL Processors P2, P3 and P4.
For redundancy, each processor shall contain two Service Processor Interfaces, a primary and a backup. A "switch" shall select one and only one of the interfaces to be active at any given point in time.

The following capabilities shall be included:

- Master Clear
- Start {Processor}
- Halt {Processor}
- Read Registers
- Write Registers
- Write Control Memory
- Read Control Memory
- Clear Selected Error

Additional information to be supplied at a later date.

2.10.3 Performance Monitoring

The performance monitoring interface shall provide the following information with respect to keypoint {See 2.6.1.7 and 2.7.1.2}:

Keypoint Code Number	4 bits
Keypoint Code	32 bits
Keypoint Data Ready	1 bit

In addition to providing keypoint data, the objectives for the performance monitoring facilities are as follows:

a. A single external device for the acquisition of performance monitoring data.

NCR/CDC PRIVATE

b. A single software system for the analysis of performance monitoring data thus acquired.

c. A single interface to both IPL Hardware and IPL Software.

d. Hardware/Firmware implementation:
Two 32-bit Counters
All events independently selectable into each counter
Exchange Package bits to enable counting
8-bit op-code and 8-bit op-code mask

e. Model-independent events (with the exception of Phase I IPL):

Number of central memory references
Number of exchange jumps
Number of page faults
Number of external interrupts
Number of Map misses
Nmmber of Segment Table references
Number of Page Table references
Number of executions of a masked op-code
Number of BDP result fields less than 9 bytes
Number of BDP result fields greater than 8 bytes
Time - 1 M hertz Free-running counter
Number of Instruction issues.
Number of central memory operand read references
Number of central memory operand write references

f. Model-dependent events:

Number of Branches out of stack
Number of Branches in stack
Number of Instruction "block" references
Number of minors waiting for central memory
Number of minors waiting for result conflict resolution
Number of Cache read references; search.
Number of Cache read references; find.
First Level/Second Level Cache monitor selection.

NCR/CDC PRIVATE

3.0 VIRTUAL MEMORY MECHANISM

3.1 General Description

IPL central memory shall be addressed by means of virtual memory addresses. This section concerns itself with the definition, formation and translation of virtual memory addresses as well as the access protection mechanisms provided in IPL systems.

3.1.1 Levels of Addresses

Within IPL systems, three levels of central memory addresses shall be recognized: Process virtual address {PVA} system virtual address {SVA}, and real memory address {RMA}.

Each process virtual address {PVA} shall consist of three major components: A segment number {SEG}, a byte number {BN} and a ring number {RN}. The process virtual address shall be local to a process and shall be translated into a global, system virtual address {SVA} by means of the process segment table. The translation process shall consist of converting the process segment number {SEG} into the system's active segment identifier {ASID} and checking the appropriate access controls to the segment.

To address central memory, the system virtual address {SVA} shall be further translated into the real memory address {RMA} through the system page table. Each paged segment shall be divided into pages and shall be allocated into real memory accordingly.

3.1.2 Address Components

The process virtual address {PVA} shall consist of a segment number {SEG}, a byte number {BN} and a ring number {RN}. The RN shall be used for access control and the combination of the SEG and BN shall specify a byte address.

The system virtual address {SVA} shall consist of an active segment identifier {ASID} and a byte number {BN}. Within the SVA, the BN shall be further divided into subfields. The BN shall consist of a page number {PN} and a page offset {PO}.

NCR/CDC PRIVATE

The concepts of segment and page are discussed in the following sub-paragraphs.

3.1.2.1 Segments

In IPL systems, data and programs shall be organized into units consisting of segments. Each segment shall be defined to be a contiguous bit-string of information with a maximum length less than or equal to 2^{31} bytes. An instruction {or datum} shall be identified {addressed} by the segment name to which it belongs and the byte name within the segment where it is located. The segment shall be defined to be the basic unit of information sharing among different processes. In order to retain a level of flexibility in naming, each process shall identify a segment with its own {process} segment number. The 12-bit process segment number shall be translated into a 16-bit system {global} segment identifier, called the active segment identifier {ASID}, by means of the process segment table. The process segment table shall effectively define the process virtual addressing space. The 12-bit process segment name shall limit the maximum number of addressable segments by a process to 4096.

The 16-bit active segment identifier {ASID} shall consist of a segment name used by the system to identify each segment currently active in the system. To each active segment, one and only one ASID shall be assigned even though it might correspond to more than one process segment number. From the perspective of the system software, the ASID shall provide a "short" name for the more permanent segment {file} name used in the IPL information storage subsystems. The translation from the permanent name to the "short" ASID shall be accomplished by the software.

3.1.2.2 Pages

To facilitate mapping segments into real memory, and to enable management of the very large central memories envisaged for the IPL, the segments shall be subdivided into pages. Page sizes shall vary between a minimum of 512 bytes and a maximum of 64k bytes. {Note: for Phase I IPL the maximum page size shall be 8k bytes}. In any given processor, the page size shall be fixed. Within each page, addressing shall be performed to the byte. The total hierarchy then, shall be:

NCR/CDC PRIVATE

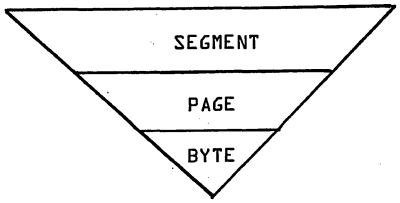
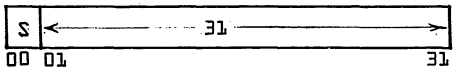


Figure 3.1-1: Address Component Hierarchy

It must be noted, however, that in general, users shall refer only to a segment and a byte number within a segment. Pages shall be transparent to the user in much the same way that central memory banks shall be transparent to users in real memory.

3.1.3 Real Memory Address

The Real Memory Address (RMA) shall be defined as a 32-bit byte address with the leftmost position referred to as the sign bit, according to the following format:



The actual central memory size shall be a system installation parameter. Addressing central memory with an RMA which exceeds the actual size of central memory, or with a negative RMA, shall generate a Real Address Out of Range interrupt.

Phase I IPL shall be size-limited to a Maximum Real Memory (RMA)

of 16 million bytes (24 bits of RMA).

3.1.4 Access Protections

Having established an environment in which many users may share code and data it is a requirement that a suitable protection mechanism be provided so as to insulate the individual users from each other. Four facilities are provided to guarantee interprocess and intraprocess protection. The interprocess protection shall be achieved via the process segment table which defines the address space of a process. The intraprocess protection shall be achieved by means of ring and key/lock facilities. Within the process address space, segments shall be organized into a privileged hierarchy according to the ring numbers associated with each of those segments. Ring one shall be the most privileged ring while ring 15 shall be the least privileged ring.

In general, a procedure executing in a particular ring shall have access to code and data in that ring and in any ring outside, (having a greater ring number than), its own. Access to inner rings can only be made through carefully controlled entry points. The key/lock facility shall be used to partition the process address space into several subspaces. In general, a procedure executing in a partition with a given key/lock shall have controlled access to the code and data of other partitions having different key/locks. When both key/lock and ring facilities are used, the process address space shall be organized with a vertical privileged hierarchy complemented by horizontal partitions.

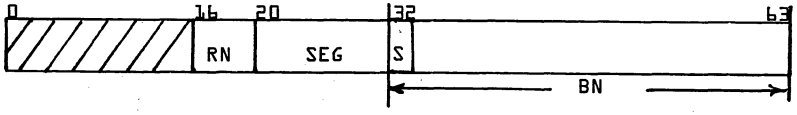
3.2 Process Virtual Address

The following paragraphs define the format of the process virtual address and the logical algorithms used for translating the process virtual address into the system virtual address.

3.2.1 Format

The process virtual address (PVA) shall constitute the effective address presented by a program (process) to address the central memory. The formation of the PVA shall be determined by the instruction repertoire and the manner in which the various fields

from each instruction shall be used to form the effective address. The format of the PVA shall be as follows:



3.2.1.1 Ring Number

The ring number (RN) shall consist of a four bit field contained in bit positions 16 through 19 of each PVA. It shall be used for access validation as discussed in section 3.6.

RN shall also be used as a special flag such that a ring number of zero, (RN = 0) shall denote an unlinked Pointer. See 2.8.1.13.

3.2.1.2 Segment Number

The segment number (SEG) shall consist of a 12-bit field contained in bit positions 20 through 31 of each PVA. It shall be used to identify a single segment from all other segments addressable by the process.

3.2.1.3 Byte Number

The byte number (BN) shall consist of a 32-bit field contained in bit positions 32 through 63 of each PVA. It shall specify the byte location (or displacement) within a segment. Bit position 32 of each PVA shall constitute the sign bit of the BN field and must be in the zero state. In the one, (negative) state, this bit shall generate an Address Specification interrupt at the time it is used to address central memory.

3.3 Process Segment Table

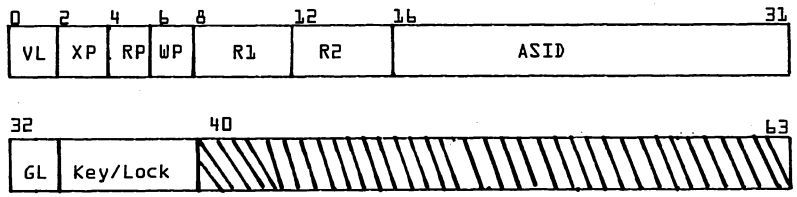
The process virtual address shall be translated into the system virtual address by means of the process segment table. The process segment table shall be specified by two values: the segment table address (STA) and the segment table length (STL). The STA shall represent the real address of the first entry of the process segment table. Each entry within a segment table shall be 64-bits long and shall be accessed by indexing the STA with the appropriate segment number. The STL, plus one, shall represent the number of usable entries in the associated segment table. The segment number, (which is applied as an index to the STA) must be less than or equal to the value of the

NCR/CDC PRIVATE

STL. The process segment table shall effectively define the process virtual address space. The maximum number of entries which may be contained in a segment table shall be 4096.

3.3.1 Segment Descriptors

Each of the 64-bit entries contained in the segment table shall be referred to as segment descriptors and shall be formatted as follows:



NCR/CDC PRIVATE

3.3.1.1 Control Fields

The 8 control bits contained in each segment descriptor, (bit positions 00 through 07), shall be grouped into 4 2-bit fields referred to as VL, XP, RP, and WP. Each of these four groups shall be decoded and translated as follows:

VL		RP	
00	Invalid Entry	00	Non-Readable Segment
01	(Reserved)	01	Read Controlled by Key/lock
10	Regular Segment	10	Read Not Controlled by Key/lock
11	Cache By-Pass Segment	11	Binding Section Segment-Read not Controlled by Key/Lock
XP		**WP**	
00	Non-Executable Segment	00	Non Writable Segment
01	Non-Privileged Executable Segment	01	Write Controlled by Key/lock
10	Local Privileged Executable Segment	10	Write not Controlled by Key/lock
11	Global Privileged Executable Segment	11	(Reserved)

Notes. Binding Section Segments shall be created by the System software (Linker) and shall be used during the execution of Call instructions as described in Section 2.6 of this specification.

Read, Write and Execution privileges are described in Section 3.6 of this specification.

3.3.1.2 Access Validation Fields

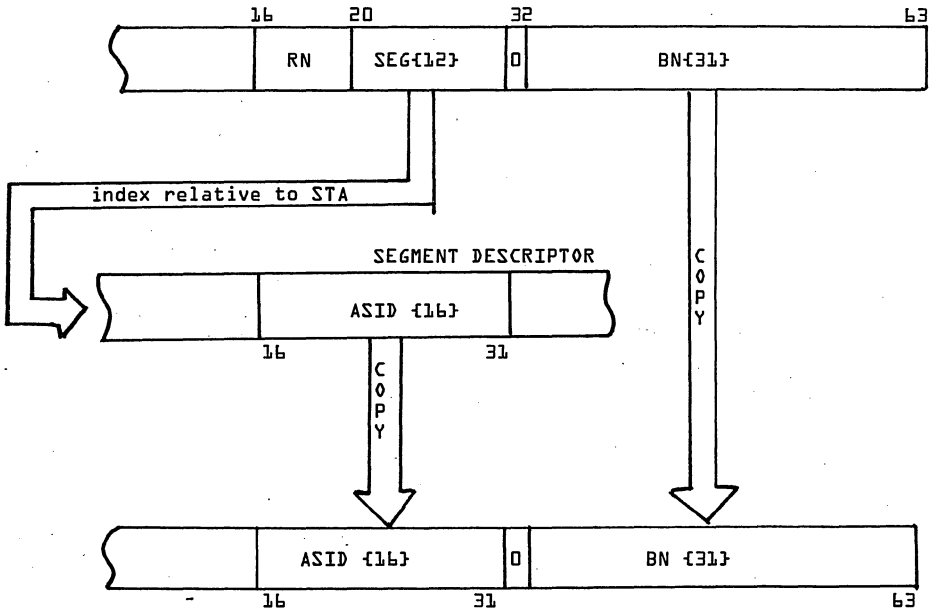
The R1 and R2 fields shall all consist of 4 bits each. GL shall be a 2-bit field and key/lock a 6-bit field. These fields shall constitute inputs to the access control mechanism in order to perform access validation as described in Section 3.6 of this specification.

3.3.1.3 Active Segment Identifier

The active segment identifier (ASID) shall consist of a 16-bit field and shall constitute a global name which identifies a single segment from all other segments currently active in the system.

3.3.1.4 Conversion to System Virtual Address

The process segment table entries shall be used primarily to validate central memory accesses. However, they shall also be utilized to convert the PVA to a system virtual address (SVA), by substituting a 16-bit active segment identifier for the 12-bit process segment number. The formation of the SVA is illustrated in Figure 3.3-1.



SYSTEM VIRTUAL ADDRESS

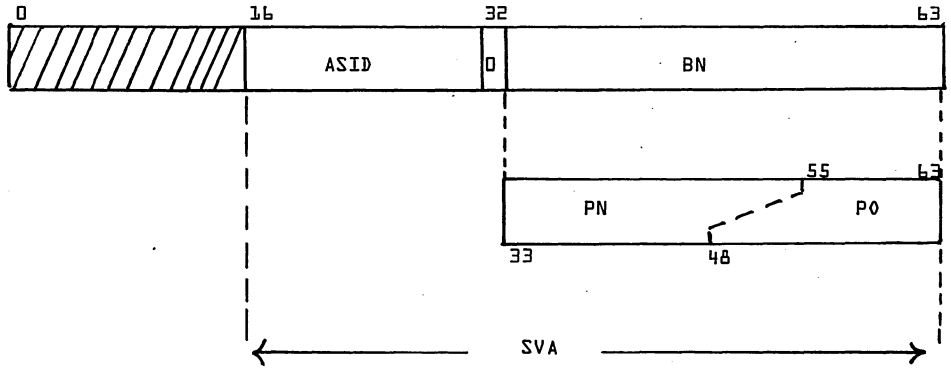
Figure 3.3-1

Conversion of PVA to SVA

NCR / CDC PRIVATE

3.4 System Virtual Address

This section specifies the format of the system virtual address and the logical algorithms used for translating the system virtual address into the real memory address. The system virtual address {SVA} shall represent a global address shared by all processes active in the system. An SVA shall consist of an active segment identifier {ASID} and a byte number {BN}. The format of an SVA shall be defined as follows:



3.4.1 Active Segment Identifier

The active segment identifier {ASID} shall consist of a 16-bit field contained in bit positions 16 through 31 of the SVA. The ASID shall represent a global name that identifies the segment from all other currently active segments in the system. Two processes which are sharing a segment may have different {process} segment numbers {SEG} to address that segment, but must have the same ASID.

3.4.2 Byte Number

The byte number {BN} shall consist of a 31-bit field contained in bit positions 33 through 63 of the SVA. It shall specify the byte location {or displacement} within a segment.

NCR / CDC PRIVATE

Within the BN, the address translation mechanism shall further recognize two subfields: a page number {PN} and a page offset {PO}.

Note. It must be stressed that these subfields are recognized only by the address translation mechanism and are transparent to general programs.

3.4.2.1 Page Number

The page number {PN} field shall be variable in size and range from 15 to 22 bits, as determined by the page size of the system. The page size shall be fixed on a per installation basis and shall not vary while the system is running. The actual size of the page number field shall be contained as a mask in the page size mask register.

3.4.2.2 Page Size Mask Register

The page size mask register shall be set such that its use against bits 48 through 54 of the SVA shall allow the separation of the page number from the page offset. Bit positions 33 through 47 of the SVA shall be automatically included in the page number, and bits 55 through 63 shall be automatically included in the page offset. The page size mask shall consist of 7 bits and shall represent a logical prefix vector with {7-U} ones, followed by U zeros, where the page size is 2^U x 512 bytes. For example, U=2 yields a page size of $2^{\{2+9\}}$=2048 bytes. The corresponding page size mask would be set to : "1111100."

3.4.2.3 Page Offset

The page offset {PO} shall represent the displacement of the central memory location to be accessed relative to the page boundary. This field shall vary with the page size and range from 9 to 16 bits.

The formation of the page number and the page offset from the byte number and the page size mask is illustrated by Figure 3.4-1 as follows:

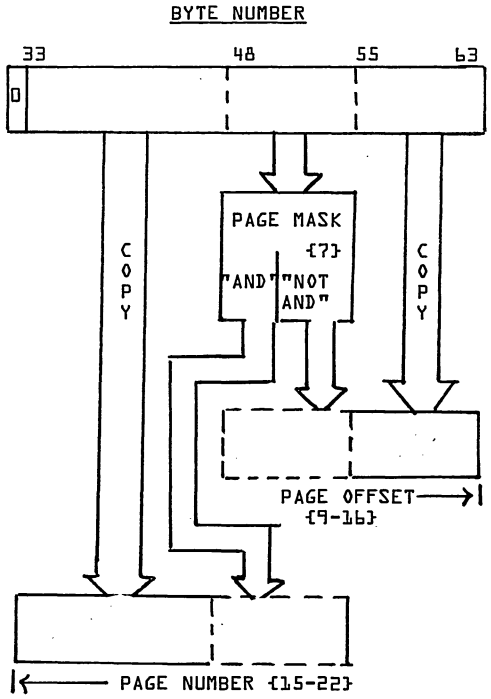


Figure 3.4-1

Formation of Page Number and Page Offset

3.5 System Page Table

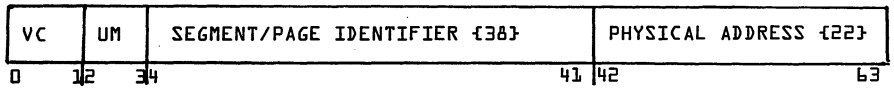
System virtual addresses shall be translated into real memory addresses by means of the system page table. The system page table shall be specified by two values: the page table address (PTA) and the page table length (PTL). The page table address shall represent the real address of the first entry of the system page table on an integral page table length boundary. Each page table entry shall consist of 64-bits. The desired entry in the page table shall be located by means of a combination of indexing and linear searching. The indexing value shall be obtained by hashing the S/PAGEID, which represents the combination of ASID and PN. Page table length shall consist of a mask which shall serve to force the index used to access the page table to be modulo the size of the table.

Page table length shall consist of 8 bits and shall also specify the length of the System Page Table as 2^n x 4096 Bytes for n=0,1, 2,...,8. The minimum page table length shall be 4096 bytes and the maximum page table length shall be 1 million bytes.

The system page table size shall be determined by real central memory size and the page size, and shall usually be 2-4 times larger than the number of available page frames.

3.5.1 Page Descriptors

System page table entries shall consist of 64-bits each, and shall be referred to as page descriptors. Each page descriptor shall identify a page frame to be accessed as well as record usage of that page frame. Page descriptors shall be formatted as follows:



NCR / CDC PRIVATE

3.5.1.1 Control Fields

The four control bits in positions 00 through 03 shall be grouped into two 2-bit fields referred to as VC and UM. These fields shall be decoded and translated as follows:

VC

00	invalid entry, stop search
01	invalid entry, continue search
10	valid entry, stop search
11	valid entry, continue search

UM

00	fresh page
01	(reserved)
10	used but not modified
11	used and modified

The search control codes shall provide the means for controlling the search of the page table for the proper SVA. They shall specify whether the page table search shall be continued to the next page descriptor when no match for the SVA is found within the current page descriptor. The used and modified codes shall indicate whether the page table entry has been used for address translation, and when used, if the real memory location was modified.

3.5.1.2 Segment/Page Identifier (S/PAGEID)

The 38-bit Segment/Page Identifier field shall identify the System Virtual Address for the Page Descriptor Entry. It shall include the 16-bit ASID and the 22-bit Page Number. For a system in which the page size is larger than 512 bytes, i.e., Page Number less than 22 bits, zeroes shall be correspondingly added in the rightmost bit positions.

3.5.1.3 Physical Address

The 22-bit Physical Address shall be used as a physical Page Frame Address.

NCR / CDC PRIVATE

When the page size is larger than 512 bytes, zeros must be present in the rightmost bit positions to obtain proper alignment. A 31-bit physical address shall be obtained as a result of address translation.

3.5.2 Allocation of Page Descriptors

3.5.2.1 Location of a Page Descriptor in the Page Table.

In order to facilitate their retrieval when necessary, a Page Descriptor associated with a given page frame is located within the System Page Table according to the following fixed procedure. There is a pseudo randomizing function {hashing function}, H, which maps a 38-bit field into a 16-bit field. This function is used to evaluate the nominal Page Descriptor location associated with a System Virtual Address {NOML{SVA}} according to the following algorithm.

NOML {SVA} = {H{ASID|PN}^{PTL|11111111 }}|0000 (See Fig. 3.5-2)

Where NOML is a 20-bit byte displacement, ASID|PN is the concatenation of ASID and PN of the System Virtual Address,

PTL is the Page Table Length Mask, "^" is a logical-AND, "|" denotes bit concatenation.

The NOML {SVA} is the displacement from the origin of the System Page Table {PTA}. NOML {SVA} identifies the first 64-bit entry which is a candidate to be associated with the specified SVA. Successive candidates' displacement shall be obtained by adding eight {modulo Page Table Length} to the previous displacement.

3.5.2.2 Search for Page Descriptor in the Page Table

The Search for Page Descriptor Algorithm accepts as a parameter of a System Virtual Address and produces as a result either a physical address pointer to the proper Page Descriptor entry or a flag indicating No Descriptor Found. The search algorithm is described in the following pseudo program:

```
Search PD : COUNT = 1
            J = NOML {SVA}   } See Figure 3.5-2 for 1st"K"
        1 : K = {PTA}| J     }
            ENTRY = Page Descriptor{K}
            if {VC = 00}^ {NOT PRE-VALIDATED} then go to 3
            if {VC = 01}^ {NOT PRE-VALIDATED} then go to 2
            if {S/PAGEID = ASID|PN} then go to 4
            if {VC = 10} then go to 3
```

Note: "NOT PRE-VALIDATED" means that the page has not been previously found in the page table as part of the current instruction's execution.

NCR/CDC PRIVATE


```
2: if {COUNT = 32} then go to 3 else COUNT = COUNT+1
   J = {J+8}^{PTL|1111111111111}, go to 1
3: Set flag "No Descriptor Found"; return
4: return {ENTRY}
```

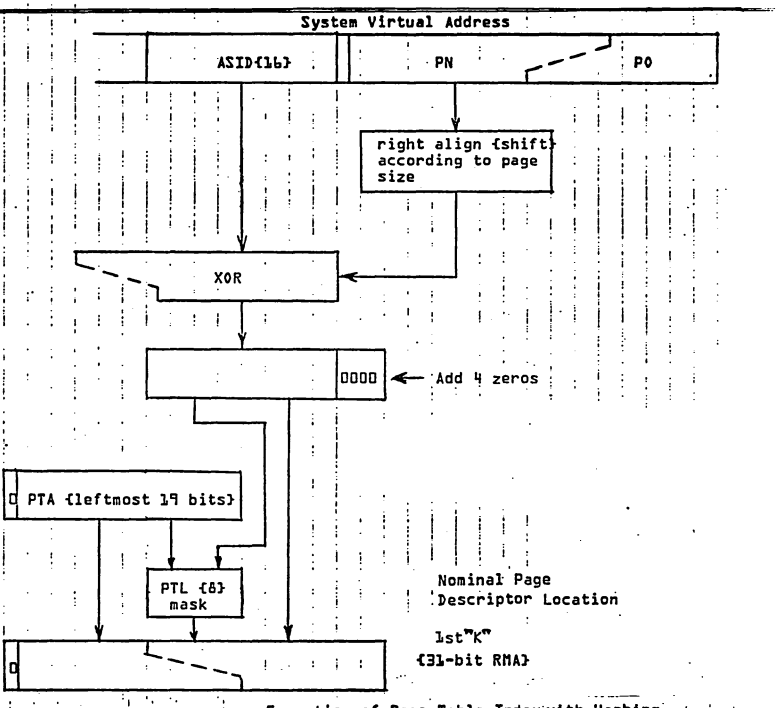


Figure 3.5-2: Formation of Page Table Index with Hashing

3.5.2.3 Formation of the Real Memory Address {RMA}

The logical algorithm for translating a system virtual address to a real memory address is depicted in Figure 3.5-1. The algorithm to obtain the proper Page Descriptor in the System Page Table has been described in the previous subparagraphs.

The dotted lines indicate the variations in field lengths which are introduced by the variable page size.

NCR/CDC PRIVATE

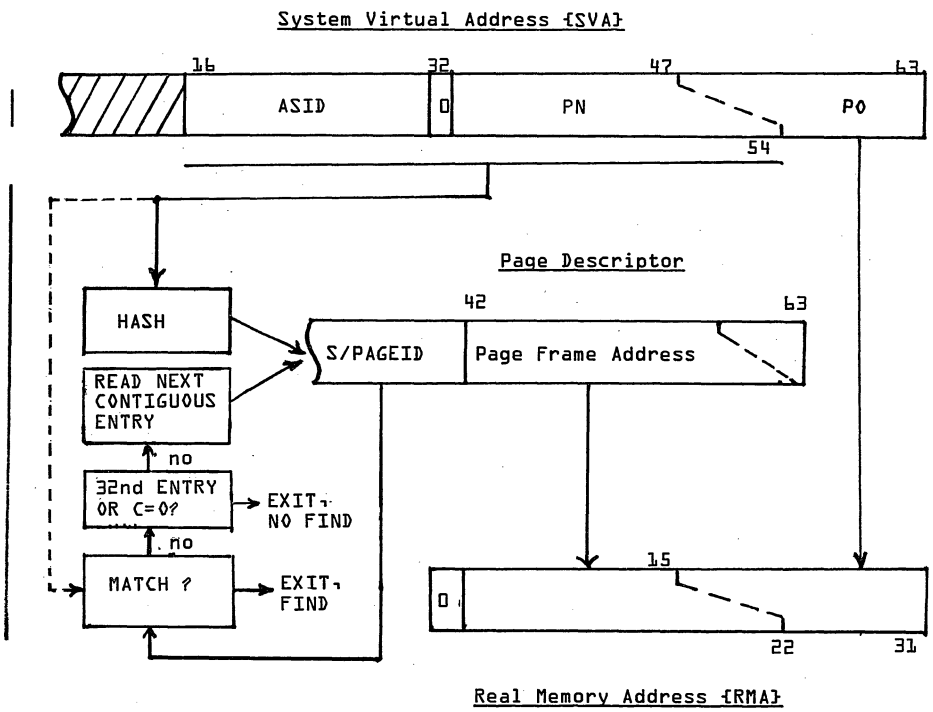


Figure 3.5-1

Translation of SVA to RMA

NCR/CDC PRIVATE

3.6 Access Protection

Within IPL, the smallest unit of access protection which can be specified shall be a segment. Four mechanisms shall be provided to facilitate interprocess and intraprocess protections. The interprocess protection shall be achieved by means of the process segment table which shall define the address space of a process. Three facilities shall be provided to achieve intraprocess protection. Segment Descriptor control fields shall be used to specify whether Read, Execute or Write access to a segment is permitted. The ring structure shall be used to organize the segments into a privileged hierarchy according to the ring number associated with each of the segments. The key/lock facility shall be used to partition the process access space into several subspaces with only restricted access from one to the other. When both ring and key/lock facilities are used, the process address space shall be organized with a vertical privileged hierarchy complemented by horizontal partitions.

3.6.1 Access Control fields.

The Execute, Read and Write access to each segment shall be controlled by the XP, RP and WP fields of each associated Segment Descriptor. The format and descriptions of the fields are specified in paragraph 3.3.1.1 of this specification.

3.6.2 Ring Hierarchy

The ability to grant access rights to a particular segment is not sufficient control, and that mechanism is augmented by a technique governing intra-process control. This technique is an extension of the common two state (system state and user state) machines. The IPL may operate in any of fifteen states. These states are rings of protection. In general, segments in the same ring have access to each other limited only by their prescribed access modes. However, communication between segments in different rings is carefully controlled. Passing control inwards (to a smaller ring number) is achieved by providing the callee with a gate through which the caller must pass. The most common example of this process occurs when a user calls on the operating system to perform a task. To ensure protection when returning from an outward call, both outward calls and inward returns shall result in interrupts and transfer of control to the operating system.

NCR/CDC PRIVATE

3.6.2.1 Execute Ring Bracket

It is frequently convenient to allow a segment to execute in several rings. This is accomplished by giving the segment an execute bracket. This bracket delimits the rings in which the segment may be executed - always provided that the segment has execute access granted by the XP field. The R1-R2 fields in the segment descriptor are used to denote the rings of which a segment may be a member.

Execute Access $R1 \leq P.RN \leq R2$

If a process is executing in a ring contained in the execute bracket of a segment, and control is transferred to that segment, then the ring of execution is unchanged, {See the Branch instruction description in sub-paragraph 2.2.3.6 of this specification}.

For the Call instruction, as described in sub-paragraph 2.6.1.2 of this specification, if the current ring of execution was greater than the ring bracket, it would be set equal to the greater ring number in the bracket, assuming the transfer of execution control is allowed.

3.6.2.2 Read and Write Limits

The concept of execute ring bracket is extended to read and write protection. A process must be executing within the read or write limit of a segment, and appropriate access must have been granted for their operations to be executed. The conditions for reading and writing a segment are given below.

Write Access	Read Access
$PVA.RN \leq R1$	$PVA.RN \leq R2$

Where the PVA.RN is the ring number contained in the A Register with which the access is being made.

3.6.2.3 Call Ring Limit {See 2.5.5 for Code Base Pointer Format}

When a procedure makes a call on another procedure executing in same or inner ring bracket, the right to make the call must first be validated, and the proper use of the gate must be checked. The

NCR/CDC PRIVATE

authority to make the call has been given to the caller if:

$PVA.RN \leq CB-R3$ {Code Base Pointer - Ring 3}

and if it is entered via the proper entry points {gate}. To further assure that the call is not made to an outer ring bracket, a check on $PVA.RN \geq R1$ is also made. The control of the call gate is implemented via the binding section which contains all the allowable entry points {code base pointers} to a procedure. The binding section is constructed by System Linker and is not modifiable by regular procedures. The format of code base pointer is described in Section 2.5.5.

3.6.3 Key/Lock Facility

The Key/Lock is another protection facility that complements the ring hierarchy for controlling the intraprocess access. It can be used to partition procedures and/or data within the same ring bracket into zones with restricted access between each other. Functionally, the Key/Lock structure is an extension of conventional storage key structure. The unit of protection, however, has been changed from physical storage blocks into virtual segments.

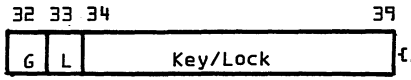
The Key/Lock facility provides the following capabilities:

- Total firewalling between subsystems in the same ring bracket.
- Total isolation of data in less privileged rings from more privileged rings.
- Facile validation of access of call arguments on calls between procedures of different keys, where one of the keys is the master key.
- Write control within a ring bracket running under the master key {e.g., process services}
- Write control of data in less privileged rings from more privileged rings.

3.6.3.1 Formats of Key/Lock fields

The 8-bit field {Bit 32 to 39} of the Segment Descriptor specifies the Key/Lock for the associated segment. The format of the field is as follows.

NCR/CDC PRIVATE



{Also, See 3.3.1}

The interpretation of the field for procedure and data segments is as follows: {When Key/Lock is equal to zero, both G and L shall be interpreted as zeroes}

Procedure

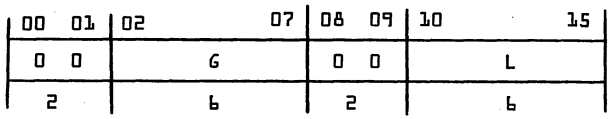
G	L		
0	0	G-master key/No Lock	L-master key
0	1	G-master key/No Lock	L-6 bit key
1	0	G-6 bit key/Lock	L-master key
1	1	G-6 bit key/Lock	L-6 bit key

Data segment

G	L		
0	0	G-No lock	L-No lock
0	1	G-No lock	L-6 bit lock
1	0	G-6 bit lock	L-No lock
1	1	G-6 bit lock	L-6 bit lock

Where G = global key/lock and L = local key/lock

Two different keys may be associated with the P-Register for the executing segment, the format is as follows.



where master key = zero
G = global key
L = local key

NCR/CDC PRIVATE

Conceptually each segment has two keys which can be tested on every access; global to global, local to local, with access being granted if both key/lock tests succeed. In fact, since there is only one six-bit field in the segment descriptor word, it is only possible for each segment to have one non-zero key so that in the case where both the global and local keys are non-zero they must be the same. The only exception to this is the P register of the machine where it is necessary to carry two non-zero non-identical keys in order to support access validation on calls and write control.

3.6.3.2 Access Validations

The key/lock is further controlled by the RP and WP controlled field in the Segment Descriptor {Please refer to Section 3.3.1.1 for the format of RP and WP}.

Read-Write access

For read or write accesses the double key comparison only occurs if key/lock control is specified for that type of access, i.e., RP = 01 and/or WP = 01.

The G-key and L-key of the P-register are tested with the G-lock and L-lock of the segment to be accessed; G to G and L to L with access being granted if both key lock tests succeed. A test is successful if: key equal to lock, or master key, or no lock.

Call/Branch {See 2.6.1.2 and 2.2.3.6, respectively}

For a Call or Branch, the P-register G-key is compared with the G-lock of the "called" or "branched to" segment. A Call or Branch is permitted if G-key equal to G-lock, or G-master key, or G-no lock.

Note: For the inter-segment Call instruction, the Key/Lock field shall be used from the Code Base Pointer only. Thus, for that instruction, the Key/Lock field from the associated segment descriptor shall be neither tested nor loaded but simply assumed to be equal to the Code Base Pointer's Key/Lock field.

P-Register

For Call/Branch, the new Global and Local keys are set according to the following rules, where

NCR/CDC PRIVATE

G-Key

Caller's G-key {Old P-register}	Callee's G-key/lock {Segment Descriptor or Code Base Pointer}	New G-key of P-register
0	0	0
0	k	k
k	0	k
k	k	k

No other key transformations are allowed.

L-key:

The new Local L-key of the P-register is always obtained from the callee's L-key in the segment descriptor or Code Base Pointer.

Return {See 2.6.1.4}

G-key {New P Register}	G-lock {Associated Segment Descriptor}	L-key {New P Register}	L-lock {Associated Segment Descriptor}
0	0	0	0
k	k	k	k
k	0		

All other key/lock combinations shall result in an Access Violation.

3.6.3.3 Software Conventions

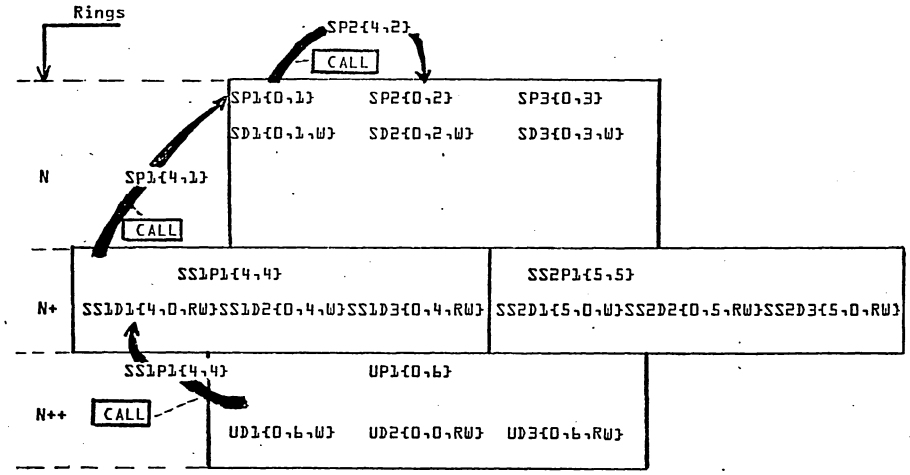
It is expected that some software conventions will be followed for using the key/lock facility. The following are examples of such conventions:

1. By using non-zero local locks, data can be restricted to be written or accessed by only "local" procedures. Normally, no procedure will have a master local key.

2. User and System procedures are normally assigned with a non-zero local key and a master global key. All non-local data are not controlled {0,0}.

3. To isolate Subsystems from each other, master global key is not assigned to them. Data to be shared by User or System {but not other Subsystems} are assigned with a global lock but no local lock. Local lock is still used for truly local data.

NCR/CDC PRIVATE



System

When called by user
SP1 can write SD1
SP1 can read SD2, SD3
SP1 can not write SD2, SD3
SP1 can not read or write UD3, SS1D3, SS2D2

When called by SS1P1
SP1 can write SD1
SP1 can read SD2, SD3
SP1 can write SS1D1
SP1 can read SS1D2
SP1 can not write SS1D2, SS2D1
SP1 can not read or write SS1D3, SS2D2, SS2D3

Note: By software convention no Procedure can have a Master Local Key

Key: SP=System Procedure
SD=System data Base
SS1=Subsystem 1
U=User
{0,1,W}={Global, Local, Access Type}
Access Type w = Write Check, No Read Check
Access Type Rw = Write Check, Read Check

Figure 3.6 : Example of Key/Lock Utilization

NCR/CDC PRIVATE

4.0 Central Memory

4.1 General

Central Memory shall provide main storage for all processors in an IPL system. It shall also provide the primary communication paths for all processors in the system. All processors shall be able to access all central memory.

Central memory may be thought of as consisting of three parts:

- Storage Units
- Distributers
- Ports

These are illustrated in Figure 4.1-1. The memory models within the IPL shall be constructed from various configurations of these elements.

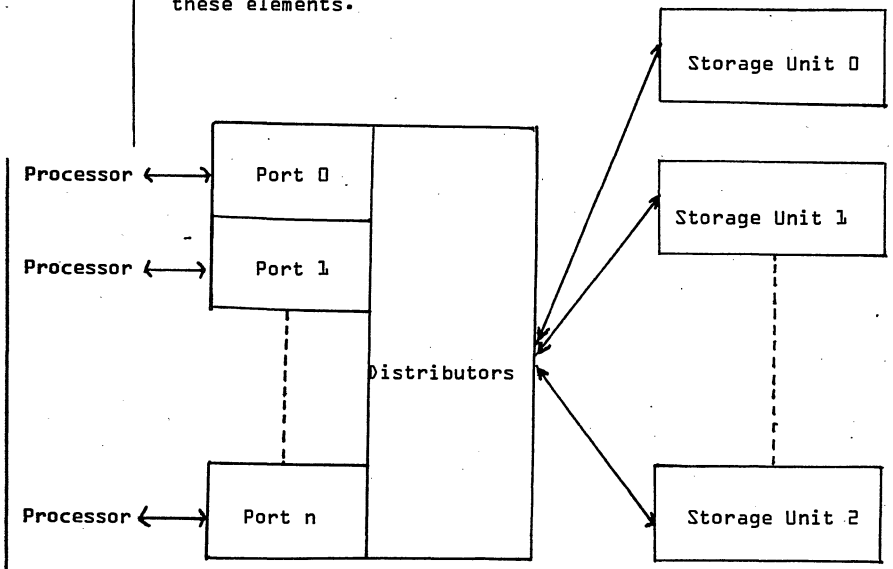


Figure 4.1-1
Memory System Elements

NCR/CDC PRIVATE

4.1.1 Memory Storage Unit

The storage unit shall be organized into eight independent banks, with each bank having a data word width of 64 bits plus 8 bits for error correcting code. The storage media shall be the 4K MOS array. The memory shall be volatile. {Information lost if power drops.}

Access to the storage unit shall have a data path 64 bits wide. Data shall be accompanied by error correcting code. Address and control signals shall be accompanied by parity bits.

Normally contiguous addresses shall be interleaved into the storage unit as illustrated in Figure 4.1-2. This shall give the storage unit a stream rate capability of one transfer per clock period. The storage unit shall also have a non-interleaved mode of operation. This mode shall allow the system to be reconfigured around a failing memory bank. This mode will degrade the stream rate for a storage unit to one transfer per bank cycle time. A more detailed explanation of this feature is contained in section 4.4.

Each storage unit shall have a maximum capacity of two million bytes, however, units shall be available in one and two megabyte options. The one mega-byte model shall be implemented by reducing the number of array paks in the storage unit by one half. The degree of interleaving within the one megabyte storage unit shall remain at eight, however, the bank size shall be reduced from 256K bytes to 128K bytes. {1K = 1024 Bytes}

NCR/CDC PRIVATE

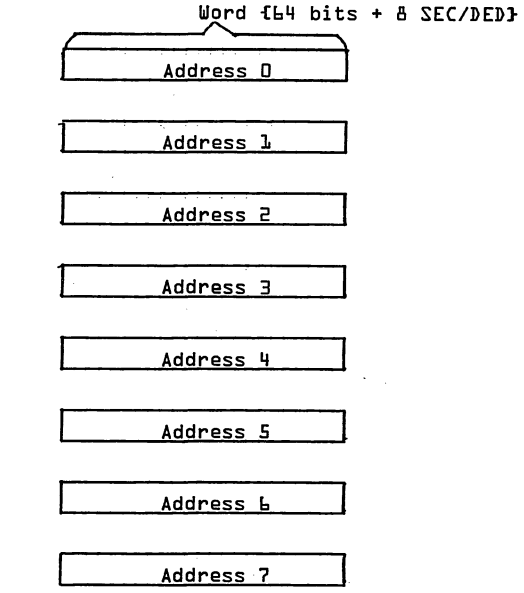


Figure 4.1-2

Memory Bank Interleaving

4.1.2 Memory Distributors

The number of central memory distributors shall differ depending on the number of ports and storage units possible on a given model. The maximum stream rate through any path shall be one word every clock cycle. Greater bandwidths may be achieved by implementing parallel distributors.

Port selection shall be made within a clock cycle. This shall allow transfers from different ports to occur at one clock cycle intervals. A port shall not request a distributor cycle if the destination bank is busy.

Port selection shall be made on a priority basis. No port shall be able to lock out any other port from a bank longer than one bank cycle time. The maximum delay that can occur at any port shall be (n-1)Tc, where n=the number of ports and Tc= bank cycle time.

Routing transfers to storage units shall be accomplished by making address translations at the distributor. Routing transfers from storage units back to ports shall be implemented with distributor generated port numbers.

Error correcting logic shall reside with the distributor hardware. This will minimize the cost of SEC/DED for the memory system. The SEC/DED hardware shall support a stream rate of one transfer per clock cycle.

The circuits required for partial write capability shall also reside within the distributor. A description of the partial write operation which requires this hardware is included in section 4.2.

Figure 4.1-3 illustrates the architecture of this distributor.

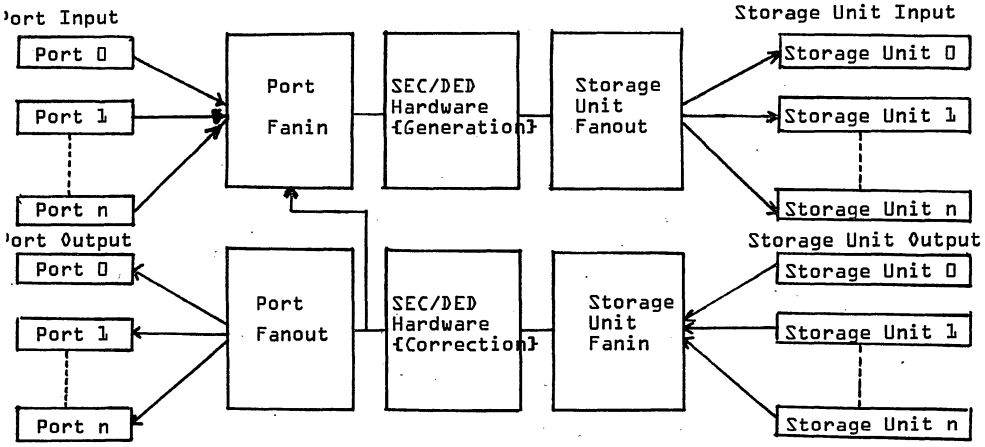


Figure 4.1-3

Memory Distributor

NCR / CDC PRIVATE

4.1.3 Memory Port

The IPL central memory shall be accessed via two types of ports. The lower IPL processor P1, as well as the Peripheral Processor, shall attach to a 32-bit port. The upper IPL processors P2, P3 and P4 shall attach to a 64-bit port. See Figures 1.3-1 through 1.3-3.
Both types of ports shall be synchronous to the hardware system clock. All processor interfaces to these ports shall also be synchronous with the hardware system clock.

Either type of port shall be capable of accepting a memory request every clock cycle. This shall give the 64 bit port a maximum bandwidth of 8 bytes per clock period, and the 32 bit port a maximum bandwidth of 4 bytes per clock period. When either port is unable to accept additional requests, due to a memory bank busy or distributor busy, it shall send a PORTBUSY signal to the processor interface. There shall be sufficient buffering within the central memory port to allow for cable delay and processor recognition of the PORTBUSY signal. This delay shall not exceed 8 clock periods.

Transmission from a processor to a central memory port shall be via coaxial cable.

Table 4.1-1 and Table 4.1-2 specify the signals at the 32 bit and 64 bit port.

NCR / CDC PRIVATE

Table 4.1-1

64 Bit Port

Input into Central Memory Port

Data In	64 lines + 8 lines {Parity}
Address	28 lines + 4 lines {Parity}
Mark Lines	8 lines + 1 line {Parity}
Tag In Lines	8 lines + 1 line {Parity}
Function Code	4 lines + 1 line {Parity}
Request	1 line

Output from Central Memory Port

Data Out	64 lines + 8 lines {Parity}
Tag Out Lines	8 lines + 1 line {Parity}
Response Code	3 lines + 1 line {Parity}
Response	1 line
Port Busy	1 line
Interrupt	1 line

Table 4.1-2

32 Bit Port

Input to Central Memory Port

Data In	32 lines + 4 lines {Parity}
Address	29 lines + 4 lines {Parity}
Mark Lines	4 lines + 1 line {Parity}
Tag In Lines	8 lines + 1 line {Parity}
Function Code	4 lines + 1 line {Parity}
Request	1 line

Output from Central Memory Port

Data Out	32 lines + 4 lines {Parity}
Tag Out Lines	8 lines + 1 line {Parity}
Response Code	3 lines + 1 line {Parity
Response	1 line
Port Busy	1 line
Interrupt	1 line

NCR/CDC PRIVATE

4.1.3.1 Port Interface Line Descriptions

a. Data In

The Data In lines shall contain the information which is to be stored into central memory during write operations. A parity bit shall accompany each byte of data.

b. Address

The address lines shall contain the address in memory which is to be accessed. The address shall be accompanied by four parity bits. The format of the address field is model dependent.

The 32 bit port shall contain one additional address bit in order to specify halfword addresses. This is required in order to perform halfword stores from the 32 bit port. The bit being clear shall specify halfword 0, the bit being set shall specify halfword 1.

c. Mark Lines

During partial write operations these lines shall indicate which bytes are valid within the 8 byte Data In word. One parity bit shall accompany the Mark Lines.

d. Tag In Lines

The Tag In lines shall contain requesting processor defined information during read or write operations. This tag information shall be returned unmodified to the requesting processor with the response from memory. The requesting processor may use this information for internal sequencing and routing of the response. A parity bit shall accompany the Tag In lines.

e. Function Code

The function lines shall contain the desired Function Code for a given memory request. Four lines shall specify up to sixteen functions. The function lines shall be accompanied by a parity bit. A detailed definition of the various functions is included in Section 4.2 of this specification.

NCR/CDC PRIVATE

f. Request

This line shall be the strobe for all signals coming into the port.

g. Data Out

The Data Out lines shall contain the information being returned in response to a read operation. A parity bit shall accompany each byte of data.

h. Tag Out Lines

The Tag Out lines shall contain a copy of the information placed on the Tag In lines during the read or write request. The requesting processor may use this information for internal sequencing and routing of the response. A parity bit shall accompany the Tag Out lines.

i. Response

This line shall be the strobe for the Data Out lines, the Tag Out lines, and the Response Code. The time interval between a request being honored and a response being returned shall be the same for all operations.

j. Response Code

These lines shall specify the nature of the response being returned to the processor. These codes are described in detail in Section 4.2. The Response Code shall be accompanied by a parity bit. Three lines specify up to eight response codes.

k. Interrupt Line

This line shall transmit an interrupt signal to the processor which is attached to the port. Section 4.2 describes how this signal is generated.

4.2 Memory Functions and Responses

4.2.1 Memory Functions

Memory shall perform the operation specified by the four bit code received on the function lines. The functions are:

NCR/CDC PRIVATE


```
0000    READ
0001    {NOT ASSIGNED}
0010    WRITE
0011    PARTIAL WRITE
0100    READ AND SET LOCK
0101    READ AND CLEAR LOCK
0110    EXCHANGE
0111    READ SYNDROME BITS
1000    READ CHECK BITS
1001    WRITE CHECK BITS
1010    READ MAINTENANCE REGISTER
1011    WRITE MAINTENANCE REGISTER
1100    INTERRUPT
1101    {NOT ASSIGNED}
1110    {NOT ASSIGNED}
1111    {NOT ASSIGNED}
```

4.2.1.1 Read

This operation shall read one central memory word from the location specified by the address received on the address lines. The normal response to a read function shall be a Read Response code, data, and the tag. Corrected data shall not be rewritten into memory.

64 Bit Port:

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Single error	Read Response Single Error, 64 bits of corrected data, tag
c. Address parity error or multiple bit error	Read Response Uncorrectable Error, 64 bits of incorrect data, tag

32 Bit Port: The 32 bit port shall have two responses. The second response shall occur one clock period after the first.

NCR/CDC PRIVATE

Condition	Response
a. Normal	1st - Read Response, 32 data bits, tag 2nd - Read Response, 32 data bits, tag
b. Single Error	1st - Read Response Single Error, 32 bits of corrected data, tag 2nd - Read Response Single Error, 32 bits of corrected data, tag
c. Address parity error or multiple bit error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag

4.2.1.2 Write

This operation shall write one central memory word with the data received on the Data In lines, at the location specified by the address received on the address lines. The normal response to a write function shall be a Write Response code and the tag. Incorrect data shall be written into memory when there is a data parity error. Writing shall be inhibited when an address parity error is detected.

64 Bit Port:

Condtion	Response
a. Normal	Write Response, tag
b. Address or data parity error	Write Response Uncorrectable Error, tag

32 Bit Port: Request, 32 bits on Data In, Function, Tag and Address Information shall be followed in one clock period with the companion 32 bits on Data In.

Condition	Response
a. Normal	Write Response, tag
b. Address or data parity error	Write Response Uncorrectable Error, tag

4.2.1.3 Partial Write

This operation shall modify one central memory word at the location specified by the address received on the address lines. Mark lines shall indicate which bytes are to be modified with the data received on the Data In lines. The normal response to a partial write function shall be a Write Response Code and the tag. This operation shall utilize a read/write cycle. Incorrect data shall be written into memory when there is a data parity error or a multiple bit error. Writing shall be inhibited when an address parity error is detected. A single bit error on read data shall be corrected before the word is modified.

64 Bit Port: Any byte within the word may be modified.

Condition	Response
a. Normal	Write Response, tag
b. Single bit error	Write Response Single Error, tag
c. Address parity error, data parity error or multiple bit error	Write Response Uncorrectable Error, tag

32 Bit Port: Request, 32 bits on Data In, Mark Lines, Function, Tag and address information shall be received in one clock period. A partial write on the 32 bit port shall not exceed four bytes. The halfword to be modified shall be specified by the address. Any combination of bytes within the halfword may be modified.

Condition	Response
a. Normal	Write Response, tag
b. Single bit error	Write Response Single Error, tag
c. Address parity error, data parity error, or multiple bit error.	Write Response Uncorrectable Error, tag

4.2.1.4 Read and Set Lock

This operation shall read and modify a central memory word at the location specified by the address received on the address lines. The logical "OR" of data read from memory and the Data In lines shall be written into memory. The normal response to

a Read and Set Lock shall be a Read Response code, data, and the tag. This operation shall utilize a read/write cycle. Writing shall be inhibited when an address parity error, data parity error, or multiple bit error is detected. Single bit errors on read data shall be corrected before the logical "or".

64 Bit Port: The bytes to be modified shall be specified by the Mark Lines.

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Single bit error	Read Response Single Error, 64 bits of corrected data, tag
c. Address parity error or multiple bit error	Read Response Uncorrectable Error, 64 bits of incorrect data, tag
d. Data parity error	Write Response uncorrectable Error, 64 bits of correct data, tag

32 Bit Port: Request, 32 bits on Data In, Mark Lines, Function, Tag, and address information shall be received in one clock period. This shall be a partial write operation for the 32 bit port. The halfword to be modified shall be specified by the address.

Condition	Response
a. Normal	1st - Read Response, 32 data bits, tag 2nd - Read Response, 32 data bits, tag
b. Single bit error	1st - Read Response Single Error, 32 corrected data bits, tag 2nd - Read Response Single Error, 32 corrected data bits, tag
c. Address parity error or multiple bit error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag
d. Data parity error	1st - Write Response Uncorrectable Error, 32 bits of correct data, tag 2nd - Write Response Uncorrectable Error, 32 bits of correct data, tag

NCR/CDC PRIVATE

4.2.1.5 Read and Clear Lock

This operation shall read and modify a central memory word at the location specified by the address received on the address lines. The logical "and" of data read from memory and the data in lines shall be written into memory. The normal response to a Read and Clear Lock shall be a Read Response, data, and the tag. This operation shall utilize a read/write cycle. Writing shall be inhibited when an address parity error data parity error, or multiple bit error is detected. Single bit errors on read data shall be corrected before the logical "and".

64 Bit Port: The bytes to be modified shall be specified by the Mark Lines.

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Single bit error	Read Response Single Error, 64 bits of corrected data, tag
c. Address parity error or multiple bit error	Read Response Uncorrectable Error, 64 bits of incorrect data, tag
d. Data parity error	Write Response Uncorrectable Error, 64 bits of correct data, tag

32 Bit Port: Request, 32 bits on Data In, Mark Lines, Function, Tag, and address information shall be received in one clock period. This shall be a partial write operation for the 32 bit port. The halfword to be modified shall be specified by the address.

Condition	Response
a. Normal	1st - Read Response 32 data bits, tag 2nd - Read Response 32 data bits, tag
b. Single bit error	1st - Read Response Single Error, 32 corrected data bits, tag 2nd - Read Response Single Error, 32 corrected data bits, tag
c. Address parity error or multiple bit error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag

NCR/CDC PRIVATE

d. Data parity error	1st - Write Response Uncorrectable Error, 32 bits of correct data, tag 2nd - Write Response, Uncorrectable Error, 32 bits of correct data, tag	

4.2.1.6 Exchange

This operation shall read and write a central memory word at the location specified by the address received on the address lines. Read data shall be sent to the requestor. Data on the Data In lines shall be written into memory. The normal response to an Exchange function shall be a Read Response Code, data, and the tag. This operation shall utilize a read/write cycle. Writing shall be inhibited when an address parity error or data parity error is detected.

64 Bit Port: The bytes to be exchanged shall be specified by the Mark Lines.

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Single bit error	Read Response Single Error, 64 corrected data bits, tag
c. Address parity error or multiple bit error	Read Response Uncorrectable Error, 64 bits of incorrect data, tag
d. Data partiy error	Write Response Uncorrectable Error, 64 bits of correct data, tag

32 Bit Port: Request, Data In, Mark Function, tag and address information shall be received in one clock period. This shall be a partial write operation and the halfword to be exchanged shall be specified by the address. Response, 32 bits of data out, and tag shall be followed (one clock period later) by Response, 2nd 32 bits of data, and tag.

Condtion	Response
a. Normal	1st - Read Response, 32 data bits tag 2nd - Read Response, 32 data bits, tag
b. Single bit error	1st - Read Response Single Error, 32 corrected data bits, tag

	2nd - Read Response Single Error, 32 corrected data bits, tag
c. Address parity error or multiple bit error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag
d. Data parity error	1st - Write Response Uncorrectable Error, 32 bits of correct data, tag 2nd - Write Response Uncorrectable Error, 32 bits of correct data, tag

4.2.1.7 Read Syndrome Bits

This operation shall read the syndrome bits which are generated as a result of reading a location specified by the address received on the address lines.

64 Bit Port: The syndrome bits shall be in byte 0 position. The remainder of the word shall be zeros.

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Address partiy error	Read Response Uncorrectable Error, 64 incorrect data bits, tag

32 Bit Port: The syndrome bits shall occupy byte 0 of halfword 0. The remainder of halfword 0 and halfword 1 shall be zeros. Response, 32 bits of data and tag shall be followed in one clock period by Response, 2nd 32 bits of data, and tag.

Conditon	Response
a. Normal	1st - Read Response, 32 data bits, tag 2nd - Read Response, 32 data bits, tag
b. Address parity error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag

4.2.1.8 Read Check Bits

This operation shall read the SEC/DED check character contained in the location specified by the address received on the address lines.

64 Bit Port: The check bits shall be in byte 0 position. The remainder of the word shall be zeros.

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Address parity error	Read Response Uncorrectable Error, 64 incorrect data bits, tag

32 Bit Port: The check bits shall occupy byte 0 of halfword 0. The remainder of halfword 0 and halfword 1 shall be zero. Response, 32 bits of data and tag shall be followed in one clock period by Response, 2nd 32 bits of data and tag.

Condition	Response
a. Normal	1st - Read Response, 32 data bits, tag 2nd - Read Response, 32 data bits, tag
b. Address parity error	1st - Read Response Uncorrectable Error, 32 bits of incorrect data, tag 2nd - Read Response Uncorrectable Error, 32 bits of incorrect data, tag

4.2.1.9 Write Check Bits

This operation shall write an eight bit check character into the location specified by the address received on the address lines. SEC/DED generation shall be disabled for this operation. The desired check character to be written shall occupy byte 0 position of the Data In lines.

64 Bit Port:

Condition	Response
a. Normal	Write Response, tag
b. Address or data parity error	Write Response Uncorrectable Error, and tag

32 Bit Port: The 32 bit port shall treat this function as a halfword write. The check character shall occupy byte 0 of halfword 0.

Condition	Response
a. Normal	Write Response, tag
b. Address or data parity error	Write Response Uncorrectable Error, and tag

4.2.1.10 Read Maintenance Register

This operation shall read the contents of the Maintenance Register specified by the address received on the address lines and return the contents along with a Read Response code to the requesting processor. A description of the Maintenance Register is included in section 4.5. This function may access the Static Status registers and the Operational Status registers. Only the Service Processor shall have access to the control registers. Any read to registers other than an existing Static Status or Operational Status shall generate an abort response code.

64 Bit Port:

Condition	Response
a. Normal	Read Response, 64 data bits, tag
b. Address parity	Read Response Uncorrectable Error 64 data bits, tag
c. Illegal Address or Function	Abort

32 Bit Port: The 32 bit port shall have two responses. The second response shall follow the first by one clock.

Conditon	Response
a. Normal	1st - Read Response, 32 data bits, tag 2nd - Read Response, 32 data bits, tag
b. Address parity Error	1st - Read Response Uncorrectable Error, 32 data bits, tag 2nd - Read Response Uncorrectable Error, 32 data bits, tag
c. Illegal address or Function	1st - Abort 2nd - Abort

4.2.1.11 Write Maintenance Register

This operation shall write the contents of the Data In lines into the Maintenance Register specified by the address received on the address lines. Only the Operational status registers may be written by the ports. A write to any other register shall generate an Abort. The normal response to a Write Maintenance Register shall be a Write Response code.

64 Bit Port:

Condition	Response
a. Normal	Write Response
b. Address parity error	Write Response Uncorrectable Error (inhibit Write)
c. Data parity error	Write Response Uncorrectable Error
d. Illegal address or Function	Abort

32 Bit Ports: Request, 32 bits on Data In, Mark Lines, Function, Tag, and Address information shall be received in one clock period. This shall be a 32 bit write operation on the 32 bit port. The halfword to be modified shall be specified by the address.

Condition	Response
a. Normal	Write Response
b. Address parity error	Write Response Uncorrectable Error

c. Data Parity Error	Write Response Uncorrectable Error
d. Illegal Address or Function	Abort

4.2.1.12 Interrupt

This function shall send an interrupt to the processor attached to the port specified by the contents of the data received on the Data In lines. Bits are assigned as follows:

```
Bit 0          Interrupt Port 0
Bit 1          Interrupt Port 1
  :                    :
  :                    :
Bit n          Interrupt Port n
```

One or more ports may receive interrupts with one interrupt operation. Any port may send an interrupt to any other port. No response is returned for this operation. The number of ports is model dependent.

The 32 bit port shall use the halfword address bit to indicate which halfword the Data In lines represent.

4.2.2 Memory Response Codes

Memory response codes shall specify the nature of the operation as the functions are processed in the memory. The response code shall be returned via the port to the sender of the function. The response codes are:

```
000    WRITE RESPONSE
001    WRITE RESPONSE UNCORRECTABLE ERROR
010    WRITE RESPONSE SINGLE ERROR
011    ABORT
100    READ RESPONSE
101    READ RESPONSE UNCORRECTABLE ERROR
110    READ RESPONSE SINGLE ERROR
111    (NOT ASSIGNED)
```

4.2.2.1 Write Response

The Write Response code shall indicate the successful completion of any write operation.

4.2.2.2 Write Response Uncorrectable Error

This code shall indicate the unsuccessful completion of any write operation.

4.2.2.3 Write Response Single Error

This code shall indicate the successful completion of a partial write operation although there was a single bit error in the data read from the location to be modified.

4.2.2.4 Abort

An Abort response shall indicate one of the following conditions:

- An attempt to reference a nonexistent address
- An illegal function
- An abnormal condition

4.2.2.5 Read Response

This response shall indicate the successful completion of one of the following operations:

- Read
  Read and Set Lock
  Read and Clear Lock
  Exchange
  Read Syndrome Bits
  Read Check Bits
  Read Maintenance Register

4.2.2.6 Read Response Uncorrectable Error

This response shall indicate the unsuccessful completion of a read operation. Read data shall be returned with the response code.

4.2.2.7 Read Response Single Error

This response shall indicate that a single error has been detected and corrected on a read operation. The corrected data shall

not be rewritten into memory.

4.3 Memory Performance Goals

4.3.1 Bandwidth

4.3.1.1 Ports

The maximum transfer rate through a memory port shall be one transfer per clock period.

4.3.1.2 Distributors

The distributors shall have independent read and write paths. The maximum transfer rate through either path shall be one transfer per clock period. Transfers on the read and write paths may occur simultaneously. In memory models with multiple distributors, read transfers may occur simultaneously with other read transfers and write transfers may occur simultaneously with other write transfers.

4.3.2 Access Time

Access time will depend to a large part on memory chip characteristics. However, it is a goal that access time as measured at the port interface, not exceed 14 clock periods.

4.3.3 Bank Cycle Time

Bank cycle time is also largely dependent on chip characteristics. It is a goal that this time not exceed 10 clock periods.

4.4 RAS Features

4.4.1 Parity

All addresses, control codes, and data which are not protected by SEC/DED shall be accompanied by parity bits. It shall be required that all processors connected to the memory have the ability to inhibit the transmission of all parity bits. This feature used in conjunction with the Operational Status registers (section 4.5) shall provide for parity network checking and fault isolation.

4.4.2 SEC/DED

All data words within the memory banks shall be protected with SEC/DED logic. It shall be possible to disable the SEC/DED hardware by utilizing the Control Registers. (Section 4.5)

Data shall be corrected on partial write operations.

The SEC/DED parity check matrix used shall minimize the chances of triple errors being interpreted as single errors.

4.4.3 Non-Interleaved Mode

A storage unit shall have the capability of operating in non-interleaved mode. This condition shall be determined by the contents of the Control Register. (Section 4.5) Placing a storage unit in non-interleaved mode shifts the bank selection bits to the high order bits of the address. The format of the address is model dependent.

The non-interleaved feature shall make sequential addresses reside in the same memory bank. This will allow software to flaw out bad addresses within a page with a minimum impact on memory capacity. The transfer rate shall be degraded to one transfer per bank cycle time.

4.5 Maintenance Registers

Memory maintenance registers shall consist of three types.

Static Status - These registers shall provide information such as equipment identification, serial number, modification history, etc.

Operation Status - These registers shall be used to reconfigure memory, disable checkers, test checkers, etc.

It shall be possible to address up to 256 maintenance registers. The three types of registers shall be mapped into the 256 possible locations consistant with the IPL processor.

The Static Status and Operational Status shall be accessible by the ports and the Service Processor interface. The Control Registers shall be accessable only by the Service Processor interface.

As specific requirements for maintenance registers become more visible, more features will be added.

NCR/CDC PRIVATE

4.5.1 Static Status

Register Address	Bit Position	Description
20	0-31	Equipment Identifier
20	32-47	Equipment Serial Number
20	48-63	Model Number
21	0-63	Modification History

4.5.2 Operational Status

Register Address	Bit Position	Description
E8	0	Indicates a single error detected
E8	1-4	Port receiving the corrected data
E8	5-12	Syndrome bits for the error
E8	13-33	Address of the failure
E8	14-63	Unassigned
E9	0	Indicates an uncorrectable error
E9	1-4	Port receiving the error
E9	5-19	These bits shall indicate the nature of the error and data path where it occurred.
E9	20-40	Specifies the memory address
E9	41-63	Unassigned

4.5.3 Control Registers

Register Address	Bit Position	Description
70	0	Disable the transmission of the parity bits on read data to the processor.
70	1	Disable SEC/DED
70	2-9	Bit indications for storage units in non-interleaved mode.
70	10	S.P. Interface Designator
70	11-16	Unassigned

The service processor interface designator bit shall be modifiable by an external switch.

NCR/CDC PRIVATE

4.6 Service Processor Interface

The service processor shall provide the fundamental signals which are required for control, maintenance, and initialization of memory.

For redundancy, each memory shall have two Service Processor Interfaces. A switch shall select one and only one of the interfaces to be active.

The specification of the Service Processor Interface is under development, however, the following capabilities are required:

- Master Clear
- Read Maintenance Registers
- Write Maintenance Registers
- Fault Line

4.7 Clock

The memory clock shall be supplied from a common system source. The clock shall have a frequency of 17.857 megahertz. All elements of the memory system shall be synchronous to this clock.

NCR/CDC PRIVATE

5.0 Hardware RAS

The scope of reliability is to reduce failure rates of hardware and software, increasing the availability of the system to the customer, and minimize component faults from becoming element and system failures. Fault-tolerant and graceful degradation techniques which yield the greatest increase in reliability per cost, and which are within the guidelines of total expenditures for RAS features shall be used.

5.1 States of Hardware

5.1.1 Fully operational - Hardware capable of rated throughput with no faults present.

5.1.2 Fault-tolerant operation - Fault has occurred, but hardware is capable of recovery from and operating with fault having no discernable impact on throughput.

5.1.3 Degraded operation - Operating with a fault occurrence and not achieving fully acceptable throughput with maintenance action in progress.

5.1.4 Down - Fault occurrence which prevents acceptable work.

5.2 Minimum Fault-tolerant and Degradable Operation Features

5.2.1 SP - The presence of another processor referred to as a service processor on all systems is required to ensure that the maintenance personnel or program can interrogate the system in the case of a failure of the central processors.

5.2.2 SEC/DED - Shall be implemented on main memory with flags to the SP and Operating System (O.S.) when error correcting and whether single error or double error has occurred.

5.2.3 Parity checking - Parity shall be checked on all data paths (1 bit/Byte), address paths, channels, and registers.

5.2.4 Degradable Cache and Map - Cache buffer and Map buffer shall have the capability of having portions of them faulted such that cache can be faulted one entry at a time and Map faulted one entry at a time.
The CPU shall also have the capability of bypassing Cache or Map or both (degraded operation).

NCR / CDC PRIVATE

5.2.5 Fault isolation - Error signals which localize faults shall be provided for O.S. and SP so that appropriate degradation or reconfiguration may take place and maintenance action time can be minimized. 90% of errors shall allow error isolation to be possible to a module level by use of hardware and software to localize a single fault. Error detection circuitry shall be designed so that errors do not propagate beyond the next interface in the system before they are detected, which will minimize the hardware checks required to localize the fault.

When an error signal is detected, diagnostics shall determine if the machine or detection circuits have failed.

5.2.6 Reconfiguration and Degradation - When permanent failures occur, the system shall be reconfigured by a combination of hardware and software techniques (goal is to be fully automatic).

All functional components (adders, busses, etc.) shall be designed with reconfiguration and degradation ideas in mind in case of failure. If an arithmetic operation such as a divide were to fail, reconfiguration in the form of subtraction could possibly be used to emulate the divide.

Reconfiguration is a way of maintaining availability by avoiding a down state. Reconfiguration/Emulation within the CPU is only possible if the error detection logic can localize the fault so that ambiguity does not exist as to what is to be reconfigured. This becomes more difficult when failures occur outside a well defined unit such as an adder with error detection, etc.

5.2.7 Instruction retry - A combination of hardware and software techniques shall be used to retry failing instructions. At a minimum, the hardware shall detect failing instructions by use

NCR / CDC PRIVATE

of error detecting circuitry (SEC/DED, parity, residue, coding, if used, etc.) and provide error signal to the O.S. and SP for software implemented retry of instructions and logging of error type that has occurred. Error signals shall cause an interrupt to an error handling routine (software). This shall occur during the failing instruction so as not to allow following instructions to alter registers or memory. The address of instruction which caused the error interrupt shall be included in the exchange package.

An error status register shall be implemented with access by SP and O.S., which will indicate what type of error occurred; such as instruction failure, memory read error, single or double error, etc.

5.2.8 Micro-Step Mode - An SP Controlled Micro Step Mode shall be implemented so as to allow micro program control instruction execution starting at any micro code address and breakpoint at any micro code address. Any number of micro instructions can then be executed, including single micro instructions.

The ECC shall also be checked on each address contents.

5.2.9 Time Out - Whenever one system facility is connected to another via command/response protocol, a time-out mechanism shall be provided to ensure continuing operation of the

NCR/CDC PRIVATE

system.

5.2.10 Power Supplies - All cabinets shall have individual power supplies, and circuit breakers shall be used in place of fuses. This philosophy will extend to all fused circuits in the machine.

5.2.11 Packaging - The number of module types shall be held to a minimum so as to reduce spares cost, increasing the likelihood of available module types on hand in the event of failure. All like modules shall be fully interchangable and replaceable when power is on.

Circuit board differences within the module (if more than one circuit card per module) shall also be held to a minimum to simplify manufacturing. Chip layout on the circuit boards shall be standardized for the purpose of reducing hole patterns to be drilled, therefore, reducing artwork layout costs and manufacturing costs.

The largest reasonable number of test points shall be provided in a standardized pattern on all modules. Access to all circuit chip pins shall be provided for probing each chip signal directly or with module extenders. Packaging and logic design shall provide the ability of using module extenders on all modules at some degraded clock speed which is unknown at this time.

5.2.12 Forced Errors - For all checking circuits (Parity, SEC/DED, etc.) and status register indications, there shall be a method of forcing conditions (programmable) so that checks can be made of the reliability circuitry.

5.2.13 Programmable Clock Margins - Clock frequency must be $\pm$ 5% program adjustable (via SP).

5.2.14 Component Failure Rates - The following page has a list of component failure rates to be used in determining the reliability of the machine. These rates are subject to change as data improves.

5.2.15 Additional Considerations - Other possible considerations are residual coding for double error detection in arithmetic units or redundant arithmetic units which may be more feasible if MSI and LSI is used, and redundancy in registers and other control areas.

NCR/CDC PRIVATE

Component Failure Rates*

Component Type	Failure Rate %/1K hrs
ECL, 10K SSI	.001
ECL, 10K MSI	.0025
ECL, 20K SSI	.004-.006
ECL, 20K MSI	.006-.008
ECL, RAMS (16 to 64 bits)	.005-.01
ECL, RAMS (256 to 1025 bits)	.01-.02
MOS, RAMS (16 to 64 bits)	.05
MOS, RAMS (256 to 1024 bits)	.08
MOS, RAMS (4K bits)	.05-.1
Terminators (Ceramic)	.0002-.001
PC BD Conn. 3500	.01-.009/Conn.
PC BD Conn. 6000	.004-.009/Conn.
PC BD Conn 7000	.01-.03/Conn.
Taper Pins	.00012/Pin
Eyelets	.0004
Wire Wraps	.000001
Conn. Pins (Cable Conn.)	.000013
8600 Inter Bd. Conn.	.00012-.00002
Circuit Boards	
3 x 3 Single Layer	.002
3 x 3 n Layers	n(.002)
5 x 8 Single Layer	.0033
5 x 8 n Layers	n(.0033)
Capacitors, TANT (Solid)	.002
Capacitors, Ceramic	.00025
Solder Joints (Thru Bd)	.00001
Solder Joints (Surface/Lap)	.0002
Wire Jumpers	.000013
Silicon Diode (Logic)	.00005

*Subject to Change as Data Improves

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
001	Load Bytes to Xk from {Aj} displaced by D and indexed by {Xi} Right, Length per S	D0→ D7	SjkiD	27
002	Load Bytes to Xk from {Aj} displaced by Q, Length per S	C0→ C7	SjkQ	27
003	Store Bytes from Xk at {Aj} displaced by D and indexed by {Xi} Right, Length per S	D8→ DF	SjkiD	27
004	Store Bytes from Xk at {Aj}, displaced by Q, Length per S	C8→ CF	SjkQ	27
005	Load Xk from {Aj} displaced by 8*D and indexed by 8*{Xi} Right	A2	jkiD	28
006	Load Xk from {Aj} displaced by 8*Q	82	jkQ	28
007	Store Xk at {Aj} displaced by 8*D and indexed by 8*{Xi} Right	A3	jkiD	28
008	Store Xk at {Aj} displaced by 8*Q	83	jkQ	28
009	Load Bytes to Xk from {Aj} displaced by D and indexed by {Xi} Right, Length per X0	A4	jkiD	29
010	Load Bytes to Xk from {Aj} displaced by Q, Length per X0	84	jkQ	29
011	Store Bytes from Xk at {Aj} displaced by D and indexed by {Xi} Right, Length per X0	A5	jkiD	29
012	Store Bytes from Xk at {Aj} displaced by Q, Length per X0	85	jkQ	29

NCR / CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
013	Load Bytes to Xk from {P} displaced by Q, Length per j	86	jkQ	30
014	Load Bit to Xk from {Aj} displaced by Q and bit indexed by {X0} Right	88	jkQ	30
015	Store Bit to Xk from {Aj} displaced by Q and bit indexed by {X0} Right	89	jkQ	30
016	Load Ak from {Aj} displaced by D and indexed by {Xi} Right	A0	jkiD	32
017	Load Ak from {Aj} displaced by Q	80	jkQ	32
018	Store Ak at {Aj} displaced by D and indexed by {Xi} Right	A1	jkiD	32
019	Store Ak at {Aj} displaced by Q	81	jkQ	32
020	Load Multiple Registers from {Aj} displaced by 8*k, Selectivity per {X0} Right	10	jk	34
021	Store Multiple Registers to {Aj} displaced by 8*k, Selectivity per {X0} Right	11	jk	34
022	Integer Sum, {Xk} replaced by {Xk} plus {Xj}	24	jk	39
023	Integer Difference, {Xk} replaced by {Xk} minus {Xj}	25	jk	40
024	Integer Product, {Xk} replaced by {Xk} times {Xj}	26	jk	40
025	Integer Quotient, {Xk} replaced by {Xk} divided by {Xj}	27	jk	41

NCR / CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
026	Integer {Xk} replaced by Absolute {Xj}	2B	jk	41
027	Integer Sum, {Xk} Right replaced by {Xk} Right plus {Xj} Right	20	jk	42
028	Integer Sum, {Xk} Right replaced by {Xj} Right plus Q	8A	jkQ	42
029	Integer Sum, {Xk} Right replaced by {Xk} Right plus j	28	jk	42
030	Integer Difference, {Xk} Right replaced by {Xk} Right minus {Xj} Right	21	jk	42
031	Integer Difference, {Xk} Right replaced by {Xk} Right minus j	29	jk	42
032	Integer Product, {Xk} Right replaced by {Xk} Right times {Xj} Right	22	jk	43
033	Integer Product, {Xk} Right replaced by {Xj} Right times Q	8C	jkQ	43
034	Integer Quotient, {Xk} Right replaced by {Xk} Right divided by {Xj} Right	23	jk	43
035	Integer Compare {Xj} to {Xk}, result to X1 Right	2D	jk	44
036	Integer Compare {Xj} Right to {Xk} Right, result to X1 Right	2C	jk	44
037	Branch to {P} displaced by 2*Q if {Xj} equal to {Xk}	94	jkQ	45
038	Branch to {P} displaced by 2*Q if {Xj} not equal to {Xk}	95	jkQ	45
039	Branch to {P} displaced by 2*Q if {Xj} greater than {Xk}	96	jkQ	46
040	Branch to {P} displaced by 2*Q if {Xj} not less than {Xk}	97	jkQ	46
041	Branch to {P} displaced by 2*Q if {Xj} Right equal to {Xk} Right	90	jkQ	46
042	Branch to {P} displaced by 2*Q if {Xj} Right not equal to {Xk} Right	91	jkQ	46
043	Branch to {P} displaced by 2*Q if {Xj} Right greater than {Xk} Right	92	jkQ	46
044	Branch to {P} displaced by 2*Q if {Xj} Right not less than {Xk} Right	93	jkQ	46
045	Branch to {P} displaced by 2*Q and increment {Xj} if {Xj} less than {Xk}	9C	jkQ	47

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
046	Branch to {P} displaced by 2*Q if SEG{Aj} not equal to SEG{Ak}; else Compare BN{Aj} to BN{Ak}, result to X1 Right	9D	jkQ	47
047	Branch to {P} indexed by 2* {Xk} Right	2E	jk	48
048	Branch to {Aj} indexed by 2* {Xk} Right	2F	jk	48.1
049	Copy to Xk from Xj	0D	jk	49
050	Copy to Xk from Aj	0B	jk	49
051	Copy to Ak from Aj	09	jk	49
052	Copy to Ak from Xj	0A	jk	49
053	Copy to Xk Right from {Xj} Right	0C	jk	49
054	Address {Ak} replaced by {Aj} plus Q	8E	jkQ	50
055	Address {Ak} replaced by {P} plus 2*{Xj} Right plus 2*Q	8F	jkQ	50
056	Address {Ak} replaced by {Ak} plus {Xj} Right	2A	jk	50
057	Enter Xk Right with plus j	3D	jk	51
058	Enter Xk Right with minus j	3E	jk	51
059	Enter Xk Right with sign extended Q	8D	jkQ	51
060	Enter X0 Right with logical jk	3F	jk	51
061	Enter Xk Left with signs per j	1F	jk	51
062	Shift {Xj} to Xk Circular, Direction and Count per {Xi} Right plus D	A8	jkiD	52
063	Shift {Xj} to Xk, Direction and Count per {Xi} Right plus D	A9	jkiD	53
064	Shift {Xj} Right to Xk Right, Direction and Count per {Xi} Right plus D	AA	jkiD	53
065	Logical Sum {Xk} replaced by {Xk} OR {Xj}	18	jk	54
066	Logical Difference, {Xk} replaced by {Xk} XOR {Xj}	19	jk	54
067	Logical Product, {Xk} replaced by {Xk} AND {Xj}	1A	jk	54
068	Logical Complement, {Xk} replaced by {Xj} Not	1B	jk	54
069	Logical Inhibit, {Xk} replaced by {Xk} AND {Xj} NOT	1C	jk	54
070	Isolate Bit Mask into Xk per {Xi} Right plus D	AC	jkiD	56
071	Isolate into Xk from Xj per {Xi} Right plus D	AD	jkiD	56
072	Insert into Xk from Xj per {Xi} Right plus D	AE	jkiD	56
073	Move Bytes Direct, {Ak} replaced by {Aj}, per X0 and X1	45	jk	57

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
074	Decimal Sum, D{Ak} replaced by D{Ak} plus D{Aj}	E0	jkQ/2	72
075	Decimal Difference, D{Ak} replaced by D{Ak} minus D{Aj}	E1	jkQ/2	72
076	Decimal Product D{Ak} replaced by D{Ak} times D{Aj}	E2	jkQ/2	72
077	Decimal Quotient, D{Ak} replaced by D{Ak} divided by D{Aj}	E3	jkQ/2	72
078	Decimal Scale, D{Ak} replaced by D{Aj} scaled per {X0} Right	E4	jkQ/2	74
079	Decimal Scale Rounded, D{Ak} replaced by rounded D{Aj} scaled per {X0} Right	E5	jkQ/2	74
080	Unimplemented Instruction			N/A
081	Unimplemented Instruction			N/A
082	Unimplemented Instruction			N/A
083	Decimal Compare, D{Aj} to D{Ak}, result to X1 Right	E6	jkQ/2	76
084	Byte Compare, D{Aj} to D{Ak}, result to X1 Right, index to X0 Right	E8	jkQ/2	77
085	Byte Compare Collated, D{Aj} to D{Ak}, both translated per {A7}, result to X1 Right, index to X0 Right	E9	jkQ/2	77
086	Byte Scan While Non-Member, D{Ak} for presence bit in D{Aj}, index to X0 Right, character to X1 Right	EA	jkQ/2	79
087	Unimplemented Instruction			N/A
088	Byte Translate, D{Ak} replaced by D{Aj}, translated per {A7}	EB	jkQ/2	80
089	Move Bytes, D{Ak} replaced by D{Aj}	EC	jkQ/2	81
090	Unimplemented Instruction			N/A
091	Byte Edit, D{Ak} replaced by D{Aj} edited per M{A7}	ED	jkQ/2	82
092	Numeric Move, D{Ak} replaced by D{Aj}, after formatting	E7	jkQ/2	75
093	Unimplemented Instruction			N/A
094	Descriptor Table Entry Move, F{k1} of D{k0, Q1} replaced by F{j1} of D{j0, Q0}	EE	jkQ/2	88

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
095	Descriptor Table Entry Increment, F{k1} of D{k0, Q1} replaced by F{k1} of D{k0, Q1} plus F{j1} of D{j0, Q0}	EF	jkQ/2	88.1
096	Calculate Subscript and Move per D{Aj} times {X0} Right, result moved to D{k0, Q1} and to X1 Right	F4	jkQ/2	91
097	Flt. Pt. Convert from Integer, Flt. Pt. {Xk} formed from Integer {Xj}	3A	jk	99
098	Flt. Pt. Convert to Integer, Integer {Xk} formed from Flt. Pt. {Xj}	3B	jk	100
099	Flt. Pt. Sum, {Xk} replaced by {Xk} plus {Xj}	30	jk	101
100	Flt. Pt. Difference {Xk} replaced by {Xk} minus {Xj}	31	jk	101
101	Flt. Pt. Sum Unnormalized, {Xk} replaced by {Xk} plus {Xj}	38	jk	101
102	Flt. Pt. Difference Unnormalized, {Xk} replaced by {Xk} minus {Xj}	39	jk	101
103	Flt. Pt. Product, {Xk} replaced by {Xk} times {Xj}	32	jk	103
104	Flt. Pt. Quotient, {Xk} replaced by {Xk} divided by {Xj}	33	jk	104
105	Flt. Pt. DP Sum {Xk, Xk+1} replaced by {Xk, Xk+1} plus {Xj, Xj+1}	34	jk	105
106	Flt. Pt. DP Difference {Xk, Xk+1} replaced by {Xk, Xk+1} minus {Xj, Xj+1}	35	jk	105
107	Flt. Pt. DP Product {Xk, Xk+1} replaced by {Xk, Xk+1} times {Xj, Xj+1}	36	jk	106
108	Flt. Pt. DP Quotient, {Xk, Xk+1} replaced by {Xk, Xk+1} divided by {Xj, Xj+1}	37	jk	107
109	Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} equal to {Xk}	98	jkQ	110
110	Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} not equal to {Xk}	99	jkQ	110
111	Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} greater than {Xk}	9A	jkQ	110
112	Branch to {P} displaced by 2*Q if Flt. Pt. {Xj} not less than {Xk}	9B	jkQ	110
113	Branch to {P} displaced by 2*Q if Flt. Pt. Exception per j contained in Xk	9E	jkQ	111
114	Compare Flt. Pt. {Xj} to {Xk}, result to X1 Right	3C	jk	112

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
115	Call per {Ai} displaced by 8*D, Arguments per {Aj}, Static Link per {Ak}	A6	jkiD	130
116	Call to {P} displaced by 2*Q, Arguments per {Aj}, Static Link per {Ak}	B0	jkQ	133
117	Return	04	jk	135
118	Pop	06	jk	136
119	Unimplemented Instruction	01	jk	N/A
120	Exchange	02	jk	139
121	Program Error	00	jk	129
122	Interrupt Processor per {Xk}	03	jk	145
123	Interrupt Product to Xk Right	1D	jk	142
124	Load Bit to Xk Right from {Aj} bit indexed by {X0} Right and set bit in Central Memory	14	jk	141
125	Compare {Xk} at {Aj}; if not equal, Load to Xk from {Aj}; if equal, Store from X0 at {Aj}	15	jk	140
126	Test Page {Aj} and Set Xk Right	16	jk	142
127	Load Page Table Index per {Xj} to Xk and Set X1 Right	17	jk	144
128	Unimplemented Instruction			N/A
129	Unimplemented Instruction			N/A
130	Copy to Xk per {Xj}	0E	jk	151
131	Copy from Xk per {Xj}	0F	jk	151
132	Copy to Xk from Central Memory Maintenance Register at {Xj} Right	08	jk	142
133	Copy to Central Memory Maintenance Register at {Xj} Right from Xk	07	jk	143
134	Branch to {P} displaced by 2*Q and alter Condition Register, per jk	9F	jkQ	147
135	Unimplemented Instruction			N/A
136	Keypoint, Class j, code equal to {Xk} Right plus Q	B1	jkQ	139
137	Unimplemented Instruction			N/A
138	Purge Buffer k of Entry per {Xj} Reserved Op Code, Model Dependent	05	jk	152
139	Execute Algorithm; Reserved Op Code, Model Dependent	B8→ BF	sjkQ	129
140	Unimplemented Instruction			N/A

NCR/CDC PRIVATE

Reference Number	Instruction Name	Operation Code	Format	Page Number
141	Unimplemented Instruction			N/A
142	Unimplemented Instruction			N/A
143	Integer Sum, {Xk} replaced by {Xj} plus Q	8B	jkQ	39
144	Compare Bytes Direct {Aj} to {Ak} per X0 and X1	44	jk	58
145	Set Xk Right per j and {X1} Right	1E	jk	58.2
146	Descriptor Field Insert from X-Register, F{k1} of D {k0, Q1} replaced by {Xj}	F0	jkQ/2	89
147	Descriptor Field Extract to X-Register, Xk replaced by F{j1} of D{j0, Q0}	F1	jkQ/2	89
148	Descriptor Field Increment by X-Register, F{k1} of D{k0, Q1} incremented by {Xj} Right	F2	jkQ/2	90
149	Descriptor Field Decrement by X-Register, F{k1} of D{k0, Q1} decremented by {Xj} Right	F3	jkQ/2	90
150	Logical Product, {Ak} replaced by {Ak} AND {Aj} per X0 and X1	42	jk	58.1
151	Logical Sum, {Ak}, replaced by {Ak} OR {Aj} per X0 and X1	40	jk	58.1
152	Logical Difference, {Ak} replaced by {Ak} EOR {Aj} per X0 and X1	41	jk	58.1
153	Move and Complement Bytes Direct, {Ak} replaced by {Aj} per X0 and X1	45 43	jk	57
154	Move Immediate Data to D{Ak}	F9	jkQ/2	91.2
155	Compare Immediate Data to D{Ak}	FA	jkQ/2	91.2
156	Add Immediate Data to D{Ak}	FB	jkQ/2	91.3
157	Load Ak with computed address per descriptor D{Aj}	FC	jkQ/2	91.4
158	Load Xk from{D{Aj}}	FD	jkQ/2	91.4
159	Store {Xj} into D{Ak}	FE	jkQ/2	91.4

NCR/CDC PRIVATE

APPENDIX B

OP	Ref.	Short Name {Format jk}	j	k	X0	X1
00	121	Program Error				
01	-	Unimplemented Instruction				
02	120	Exchange				
03	122	Interrupt Processor		X		
04	117	Return				
05	138	Purge	X	Sub-op		
06	118	Pop				
07	133	Copy to C.M. Maint. Register	X	X		
08	132	Copy from C.M. Maint. Register	X	X		
09	051	Copy to Ak from Aj	A	A		
0A	052	Copy to Ak from Xj	X	A		
0B	050	Copy to Xk from Aj	A	X		
0C	053	Copy to Xk from Xj, Halfword	X	X		
0D	049	Copy to Xk from Xj	X	X		
0E	130	Copy from State Register	X	X		
0F	131	Copy to State Register	X	X		
10	020	Load Multiple Registers	A	Displace	Select	
11	021	Store Multiple Registers	A	Displace	Select	
12		Unimplemented Instruction				
13		Unimplemented Instruction				
14	124	Test and Set, Bit	A	X	Index	
15	125	Compare and Swap	A	X	Lock	
16	126	Test Page	A	X		
17	127	Load Page Table Index	X	X		Mark
18	065	Logical Sum	X	X		
19	066	Logical Difference	X	X		
1A	067	Logical Product	X	X		
1B	068	Logical Complement	X	X		
1C	069	Logical Inhibit	X	X		
1D	123	Logical Product/Interrupt Mask		X		
1E	145	Convert Mark to Boolean	Sub-op	X		Argument
1F	061	Extend Halfword Sign per j	Sub-op	X		

NCR/CDC PRIVATE

APPENDIX B

Op	Ref.	Short Name {Format jk}	j	k	X0	X1
20	027	Add Integer, Halfword	X	X		
21	030	Subtract Integer, Halfword	X	X		
22	032	Multiply Integer, Halfword	X	X		
23	034	Divide Integer, Halfword	X	X		
24	022	Add Integer	X	X		
25	023	Subtract Integer	X	X		
26	024	Multiply Integer	X	X		
27	025	Divide Integer	X	X		
28	029	Increase Halfword by j	Operand	X		
29	031	Decrease Halfword by j	Operand	X		
2A	056	Add Address, Halfword	X	A		
2B	026	Convert to Absolute	X	X		
2C	036	Compare Integer, Halfword	X unless 0	X unless 0		Mark
2D	035	Compare Integer	X unless 0	X unless 0		Mark
2E	047	Unconditional Branch, Intra-Segment	-	X		
2F	048	Unconditional Branch, Inter-Segment	A	X		
30	099	Add Flt. Pt.	X	X		
31	100	Subtract Flt. Pt.	X	X		
32	103	Multiply Flt. Pt.	X	X		
33	104	Divide Flt. Pt.	X	X		
34	105	Add Flt. Pt., Double Precision	X {2}	X {2}		
35	106	Subtract Flt. Pt., Double Precision	X {2}	X {2}		
36	107	Multiply Flt. Pt., Double Precision	X {2}	X {2}		
37	108	Divide Flt. Pt., Double Precision	X {2}	X {2}		
38	101	Add Flt. Pt., Unnormalized	X	X		
39	102	Subtract Flt. Pt., Unnormalized	X	X		
3A	097	Convert Integer to Flt. Pt.	X	X		
3B	098	Convert Flt. Pt. to Integer	X	X		
3C	114	Compare Flt. Pt.	X unless 0	X unless 0		Mark
3D	057	Enter Halfword with plus j	Operand	X		
3E	058	Enter Halfword with minus j	Operand	X		
3F	060	Enter Halfword with plus jk	Operand		Dest. Reg.	
40	151	Logical Sum, Direct	A	A	Fill/Length	Mark/Length
41	152	Logical Difference, Direct	↓	↓	↓	↓
42	150	Logical Product, Direct				
43	153	Move Complement, Direct				
44	144	Compare, Direct				
45	073	Move, Direct				
46 ↓ 7F	- ↓	Unimplemented Instruction ↓				

NCR/CDC PRIVATE

APPENDIX B

Op	Ref.	Short Name {Format jkQ}	j	k	X0	X1
80	017	Load A Bytes, Displaced	A	X		
81	019	Store A Bytes, Displaced	A	X		
82	006	Load X Word, Displaced	A	X		
83	008	Store X Word, Displaced	A	X		
84	010	Load X Bytes, Displaced	A	X	Length	
85	012	Store X Bytes, Displaced	A	X	Length	
86	013	Load X Bytes, Displaced Relative	Length-1	X		
87	-	Unimplemented Instruction				
88	014	Load X, Bit	A	X	Index	
89	015	Store X, Bit	A	X	Index	
8A	028	Add Integer, Halfword plus Q	X	X		
8B	143	Add Integer, Word plus Q	X	X		
8C	033	Multiply Integer, Halfword times Q	X	X		
8D	059	Enter Halfword with Q	-	X		
8E	054	Add Address, A plus Q	A	A		
8F	055	Add Address, X plus P plus Q	X unless 0	A		
90	041	Branch EQ, Halfword Integer	X unless 0	X unless 0		
91	042	Branch NE, Halfword Integer	↓	↓		
92	043	Branch GT, Halfword Integer				
93	044	Branch Not LT, Halfword Integer				
94	037	Branch EQ, Integer				
95	038	Branch NE, Integer				
96	039	Branch GT, Integer				
97	040	Branch Not LT, Integer				
98	109	Branch EQ, Flt. Pt.				
99	110	Branch NE, Flt. Pt.				
9A	111	Branch GT, Flt. Pt.				
9B	112	Branch Not LT, Flt. Pt.				
9C	045	Branch and Increment LT, Integer	X	X		Mark
9D	046	Branch NE, SEG; else Compare BN	A	A		
9E	113	Branch EQ, Flt. Pt. Exception	Sub-op	X		
9F	134	Branch/Alter, Condition Register	Bit	Sub-op		

NCR / CDC PRIVATE

APPENDIX B

Op	Ref.	Short Name {Format jkiD}	j	k	i	X0
A0	016	Load A Bytes, Indexed/Displaced	A	A	X unless 0	
A1	018	Store A Bytes, Indexed/Displaced	A	A	↓	
A2	005	Load X Word, Indexed/Displaced	A	X		
A3	007	Store X Word, Indexed/Displaced	A	X		
A4	009	Load X Bytes, Indexed/Displaced	A	X		Length
A5	011	Store X Bytes, Indexed/Displaced	A	X		Length
A6	115	Call, Displaced	A	A	A	Select
A7	-	Unimplemented Instruction				
A8	062	Shift Word, Circular	X	X	X unless 0	
A9	063	Shift Word, End-off	X	X	↓	
AA	064	Shift Halfword, End-off	X	X		
AB	-	Unimplemented Instruction				
AC	070	Isolate Bit Mask	-	X	X unless 0	
AD	071	Isolate Bit String	X	X	↓	
AE	072	Insert Bit String	X	X		
AF	-	Unimplemented Instruction				

Op	Ref.	Short Name {Format jkQ}	j	k	X0
B0	116	Call, Displaced Relative	A	A	Select
B1	136	Keypoint	Code	X, unless 0	
B2	-	Unimplemented Instruction			
↓ B7		↓			

Op	Ref.	Short Name {Format SjkQ}	j	k
B8	139	Execute Algorithm 0	-	-
↓		↓ ↓	-	-
BF		?	-	-
C0	002	Load X Bytes, Displaced L=1	A	X
↓		↓ ↓	A	X
C7		L=8	A	X
C8	004	Store X Bytes, Displaced L=1	A	X
↓		↓ ↓	A	X
CF		L=8	A	X

Op	Ref.	Short Name {Format SjkiD}	j	k	i
D0	001	Load X Bytes, Indexed/Displaced, L=1	A	X	X unless 0
↓		↓ ↓	A	X	↓
D7		L=8	A	X	
D8	003	Store X Bytes, Indexed/Displaced, L=1	A	X	
↓		↓ ↓	A	X	
DF		L=8	A	X	

NCR / CDC PRIVATE

APPENDIX B

Op	Ref.	Short Name (Format jk2/2)	j0	j1	k0	k1	A7	X0	X1
E0	074	BDP Numeric Sum	*	**	*	**			
E1	075	Difference	↓	↓	↓	↓			
E2	076	Product	↓	↓	↓	↓			
E3	077	Quotient	↓	↓	↓	↓			
E4	078	Scale	↓	↓	↓	↓		S.C.	
E5	079	Scale Rounded	↓	↓	↓	↓		S.C.	
E6	083	Compare	↓	↓	↓	↓			Mark
E7	092	Move	↓	↓	↓	↓			
E8	084	Byte Compare	↓	↓	↓	↓		Count	Mark
E9	085	Compare Collated	↓	↓	↓	↓	Table	Count	Mark
EA	086	Scan While Non-Member	↓	↓	↓	↓		Count	Find
EB	088	Translate	↓	↓	↓	↓	Table		
EC	089	Move	↓	↓	↓	↓			
ED	091	Edit	↓	↓	↓	↓	Pattern		
EE	094	Descriptor Move	↓	***	↓	***			
EF	095	Increment		***					
F0	146	Insert		X					
F1	147	Extract	*	***		X			
F2	148	Increment by X		X	*	***			
F3	149	Decrement by X		X	↓	***			
F4	096	Calculate Subscript	*	**	↓	—		Mult.	Dest.
F5	150	Unimplemented Instruction							
F6	151	↓							
F7	152								
F8	153								
F9	154	Immediate Data Move	Sub-op		*	* *			
FA	155	Compare	↓		↓	↓			Mark
FB	156	Add	↓		↓	↓			
FC	157	Load A-Register, Computed Address	*	**	A				
FD	158	Load X-Register, BDP Data	*	**	X				
FE	159	Store X-Register, BDP Data		X	*	**			
FF	-	Unimplemented Instruction							

Mult: Multiplier
Dest: Destination

* : A4/A5 Selection
** : A8 → AF Selection
*** : Data Descriptor field designator
S.C. : Shift Count

NCR / CDC PRIVATE