Very Rare

Copy of
CDC 8600
Manual

Gordon Bell
BARC, Suite 830
301 Howard Street
San Francisco CA 94105-2241
USA

Gordon,
Herewith

8600 Ref Man (Prel)
US Patent 3533889

Regards

Rob Pike

Job Code	Sheet Title	SYDAC
		Sheet Ref. A4

Issue	Changes Since Last Issue	Originator .. Date	Checked	Approved
		../../..	../../..	../../..
		../../..	../../..	../../..
		../../..	../../..	../../..

FILE X.2

8600
COMPUTER SYSTEM

8600 Ref. Manual

CONTROL DATA CORPORATION
CHIPPEWA LABORATORY

CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION *
ONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION *
NTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * C
TROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CH
ROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHI
OL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIP
L DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPP
DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPE
DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEW
ATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA
TA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA
A CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA L
CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LA
CORPORATION * C * CHIPPEWA LAB
ORPORATION * CH CHIPPEWA LABO
RPORATION * CHI CHIPPEWA LABOR
PORATION * CHIP HIPPEWA LABORA
ORATION * CHIPP PRELIMINARY IPPEWA LABORAT
RATION * CHIPPE PPEWA LABORATO
ATION * CHIPPEW 8500 PEWA LABORATOR
TION * CHIPPEWA EWA LABORATORY
ION * CHIPPEWA REFERENCE MANUAL WA LABORATORY
ON * CHIPPEWA L A LABORATORY *
N * CHIPPEWA LA LABORATORY *
* CHIPPEWA LAB LABORATORY * C
* CHIPPEWA LABO ABORATORY * CO
CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CON
CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONT
HIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTR
IPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTRO
PPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL
PEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL
EWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL D
WA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DA
A LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DAT
LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA
LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA
ABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA C
BORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CO
ORATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA COR
RATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORP
ATORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPO
TORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPOR
ORY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORA
RY * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORAT
Y * CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATI
* CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATIO
* CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION
CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION
CONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION *
ONTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION *
NTROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * C
TROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CH
ROL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHI
OL DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIP
L DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPP
DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPE
DATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEW
ATA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA
TA CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA
A CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA L
CORPORATION * CHIPPEWA LABORATORY * CONTROL DATA CORPORATION * CHIPPEWA LA

SPECIAL COPY FOR CANADIAN DIV.
****

PRELIMINARY

8500

REFERENCE MANUAL

This is an incomplete working draft and will be enlarged, corrected and revised periodically. Distribution is restricted to those listed below.

Distribution:

L. W. Gallup
R. C. Moore
T. A. Ammerman
D. W. Swedberg
D. A. Canlander
M. C. Willis
P. G. Juetten
G. Mc Crossen
R. W. Sauls
D. G. Grina
W. L. Bhend
R. A. Horton
J. B. Pearson
W. C. Katz
G. A. Simpson

11-03-72

Table of Contents

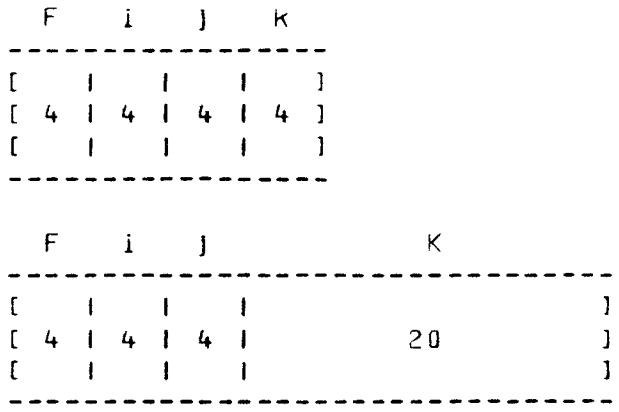
B

Part 1

8080 INSTRUCTION FORMATS & CODES

INSTRUCTION FORMATS

4-bit instruction codes (1 or 2 parcels)

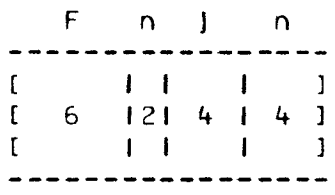


F = Instruction code

i, j, k = X register designators for operand source and destination

K = 20 bit constant

6-bit instruction codes (1 parcel)

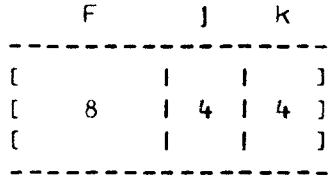


F = Instruction code

n = 6-bit constant

j = X register designator for operand source

8-bit instruction codes (1 or 2 parcels)



F = Instruction code

j,k = X register designators for operand source and destination

K = 20-bit program constant

8000 INSTRUCTION CODES

dibits/hex.			clock periods
0000	00	Program error exit	
0001	01	Logical product of (Xj) and (Xk) to Xj	2
0002	02	Logical sum of (Xj) plus (Xk) to Xj	2
0003	03	Logical difference of (Xj) minus (Xk) to Xj	2
0010	04	Copy (Xk) to Xj	2
0011	05	Copy complement of (Xk) to Xj	2
0012	06	Shift (Xj) left by (Xk) or right if (Xk) is negative	3
0013	07	Shift (Xj) right by (Xk) or left if (Xk) is negative	3
0020	08	Floating DP sum of (Xj) plus (Xk) to Xj	8
0021	09	Floating DP difference of (Xj) minus (Xk) to Xj	8
0022	0A	Floating divide of (Xj) by (Xk) to Xj	32
0023	0B	Population count of (Xk) to Xj	5
0030	0C	Floating DP product of (Xj) times (Xk) to Xj	8/3
0031	0D	Integer product of (Xj) times (Xk) to Xj	8/3
0032	0E	Program error exit	
0033	0F	Pass	1

dibits/hex.			clock periods
0100	10	Transmit k to Xj	2
0101	11	Transmit complement k to Xj	2
0102	12	Integer sum of (Xj) plus k to Xj	3
0103	13	Integer difference of (Xj) minus k to Xj	3
0110	14	Unpack coefficient of (Xj) to Xk	2
0111	15	Unpack exponent of (Xj) to Xk	2
0112	16	Pack coefficient Xk and exponent Xj to Xk	2
0113	17	Integer difference 0 - (Xk) to Xj	3
0120	18	Begin system call [MTF]	1
0121	19	End system call [MTF]	1
0122	1A	Block read input channel (Xj) to address (Xk) [MTF]	
0123	1B	Block write output channel (Xj) from addr.(Xk) [MTF]	
0130	1C	Read channel request to Xj [MTF]	4
0131	1D	Enter XA from Xk [MTF]	1
0132	1E	Program error exit	
0133	1F	Program error exit	

dibits/hex.			clock periods
0200	20	Store (Xj) data into memory at address K	1
0201	21	Store (Xj) data into memory at address (Xk)	1
0202	22	Read/store: data at addr. K to Xj / (Xj) to addr. K	15+
0203	23	Read/store: data at addr.(Xk) to Xj / (Xj) to addr.(Xk)	15+
0210	24	Read data at address K to Xj	15+
0211	25	Read data at address (Xk) to Xj	15+
0212	26	Read program at absolute address K to Xj	15+
0213	27	Read program at absolute address (Xk) to Xj	15+
0220	28	Read program at absolute address P + K to Xj	15+
0221	29	Transmit P + K to Xj	3
0222	2A	Transmit K to Xj	2
0223	2B	Transmit XA to Xj	2
0230	2C	Set interlock flags from Xk (IPF)	1
0231	2D	Clear interlock flags from Xk (IPF)	1
0232	2E	Read interlock register to Xj	2
0233	2F	Read internal clock to Xj	2

dibits/hex.			clock periods
0300	30	Jump to P + K	7-18+
0301	31	Set (Xj) = P and call subroutine at P + K	7-18+
0302	32	Jump to P + K if (Xj) in range	3-7-18+
0303	33	Jump to P + K if (Xj) not in range	3-7-18+
0310	34	Jump to P + K if (Xj) is equal to zero	3-7-18+
0311	35	Jump to P + K if (Xj) is not equal to zero	3-7-18+
0312	36	Jump to P + K if (Xj) is positive	3-7-18+
0313	37	Jump to P + K if (Xj) is negative	3-7-18+
0320	38	Set (Xj) = P and call subroutine at K	7-18+
0321	39	Set (Xj) = P and call subroutine at (Xk)	7-18+
0322	3A	Set (Xj) = P & call lib. routine at addr. K [clear PRF]	18+
0323	3B	Set (Xj) = P & call lib. rout. at addr.(Xk) [clear PRF]	18+
0330	3C	Subroutine exit to (Xj) + K	7-18+
0331	3D	Library exit to (Xj) + K [set/clear PRF]	18+
0332	3E	Jump to K	7-18+
0333	3F	Exchange exit	

dibits/hex.			clock periods
100X	40	Save lower (Xj) for n bits	2
101X	44	Blank lower (Xj) for n bits	2
102X	48	Left shift (Xj) by n bits	3
103X	4C	Right shift (Xj) by n bits	3
11XX	5x	Integer sum of (Xj) plus K to Xi	3
12XX	6x	Integer sum of (Xj) plus (Xk) to Xi	3
13XX	7x	Integer difference of (Xj) minus (Xk) to Xi	3
20XX	8x	Floating sum of (Xj) plus (Xk) to Xi	8
21XX	9x	Floating difference of (Xj) minus (Xk) to Xi	8
22XX	Ax	Floating product of (Xj) times (Xk) to Xi	8/3
23XX	Bx	Branch backward i words if (Xj) < (Xk)	3-7-18+
30XX	Cx	Read data at address (Xj) + K to Xi	15+
31XX	Dx	Read data at address (Xj) + Xk to Xi	15+
32XX	Ex	Store data at address (Xj) + K from Xi	1
33XX	Fx	Store data at address (Xj) + (Xk) from Xi	1

Part 2

SYSTEM DESCRIPTION

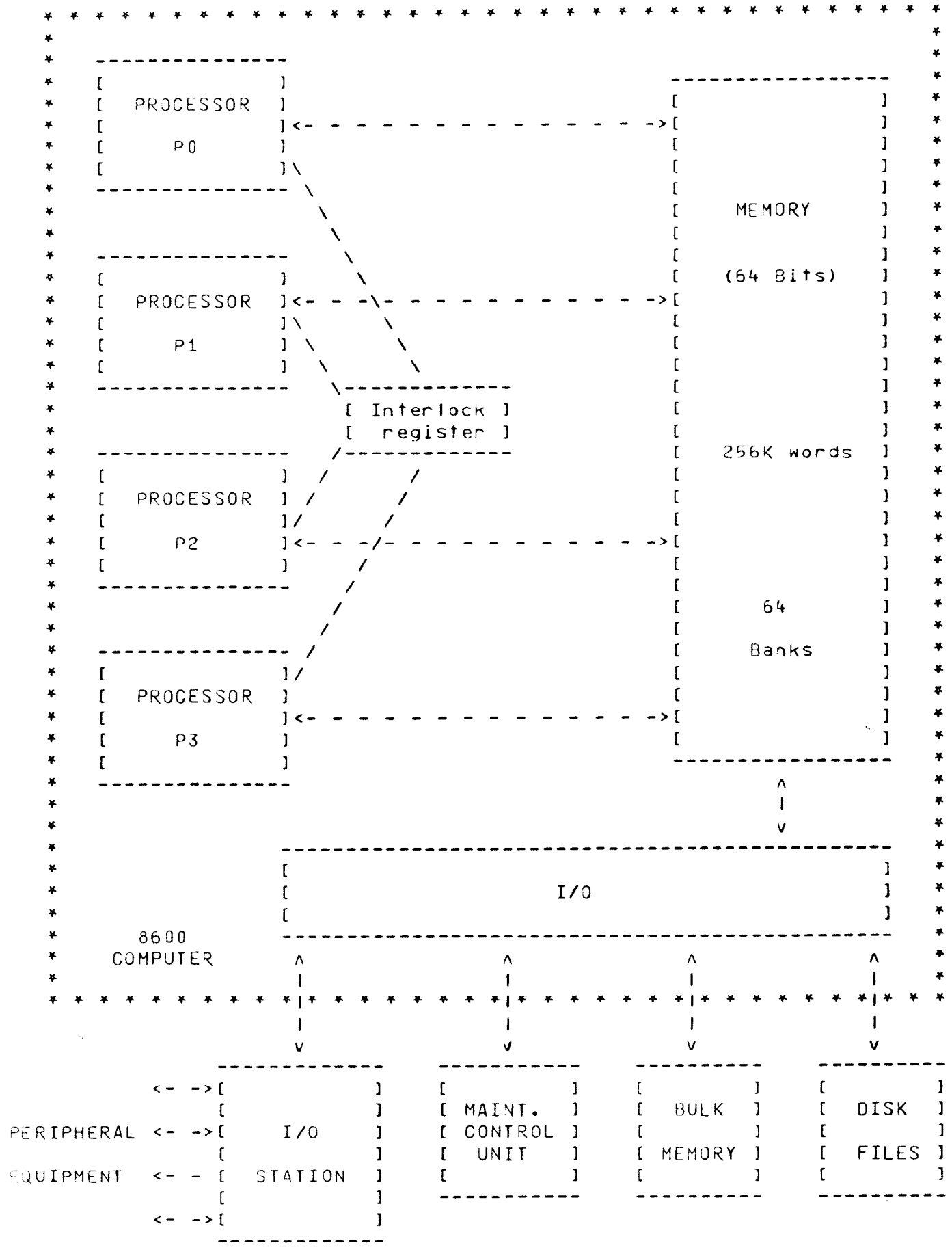


Fig. 2-1 8600 SYSTEM

B

INTRODUCTION

The 8600 system is a multi-processor system with four 8000 processors sharing a common 256K 64-bit word memory (figure 2-1). The processors communicate with an I/O station, maintenance control unit (MCU), bulk memory, and disk files through the common memory via sixteen independent I/O channels. The I/O station (typically a 6000 station configuration or a 7000 I/O station) handles all I/O operations.

Each of the four 8000 processors is an independent computation unit including arithmetic units, sixteen 64-bit operating registers, and a twelve word instruction stack (figure 2-2). Part 1 of this manual lists the instruction repertoire for the 8000 processors, and Part 3 provides descriptive information for each instruction. Each processor executes programs or program segments stored in the common memory upon command or assignment by the operating system software or programs operating under the control of the operating system. The 8600 operating system software will be described in a separate manual.

8600 System Parameters

8000 Processor Unit (13 modules)

- 64-bit internal word
- binary computation in fixed point and floating point format
- twelve word instruction stack
- synchronous internal logic with 8 nanosecond clock period (clock frequency variable in 5% increments by MCU program)

Memory (66 modules)

- 256K words of linear select memory (64-bit words)
- 64 independent banks
- 4096 words per bank
- 250? nanosecond read/write cycle time
- 8 nanosecond per word maximum transfer rate

I/O Section (8 modules)

- 16 channels
- each channel full duplex
- 40? nanosecond per 64-bit word maximum transfer rate

The 8600 system is the result of a development program to provide computing capacity substantially beyond that of the 7600 systems. By using the multi-processing capabilities of the 8600, computation in the 8600 is expected to average ten times as fast as corresponding computation in the 7600 system. The 8600 is not machine code compatible with 7600 systems.

The 8600 is physically packaged in 125 pluggable modules (6" x 8" x 2.5"). There are 13 modules in each of the four 8000 processors, 66 modules in the memory, and 8 I/O modules. In the manual sections which follow, the system operation will be described in terms of module functions when practicable.

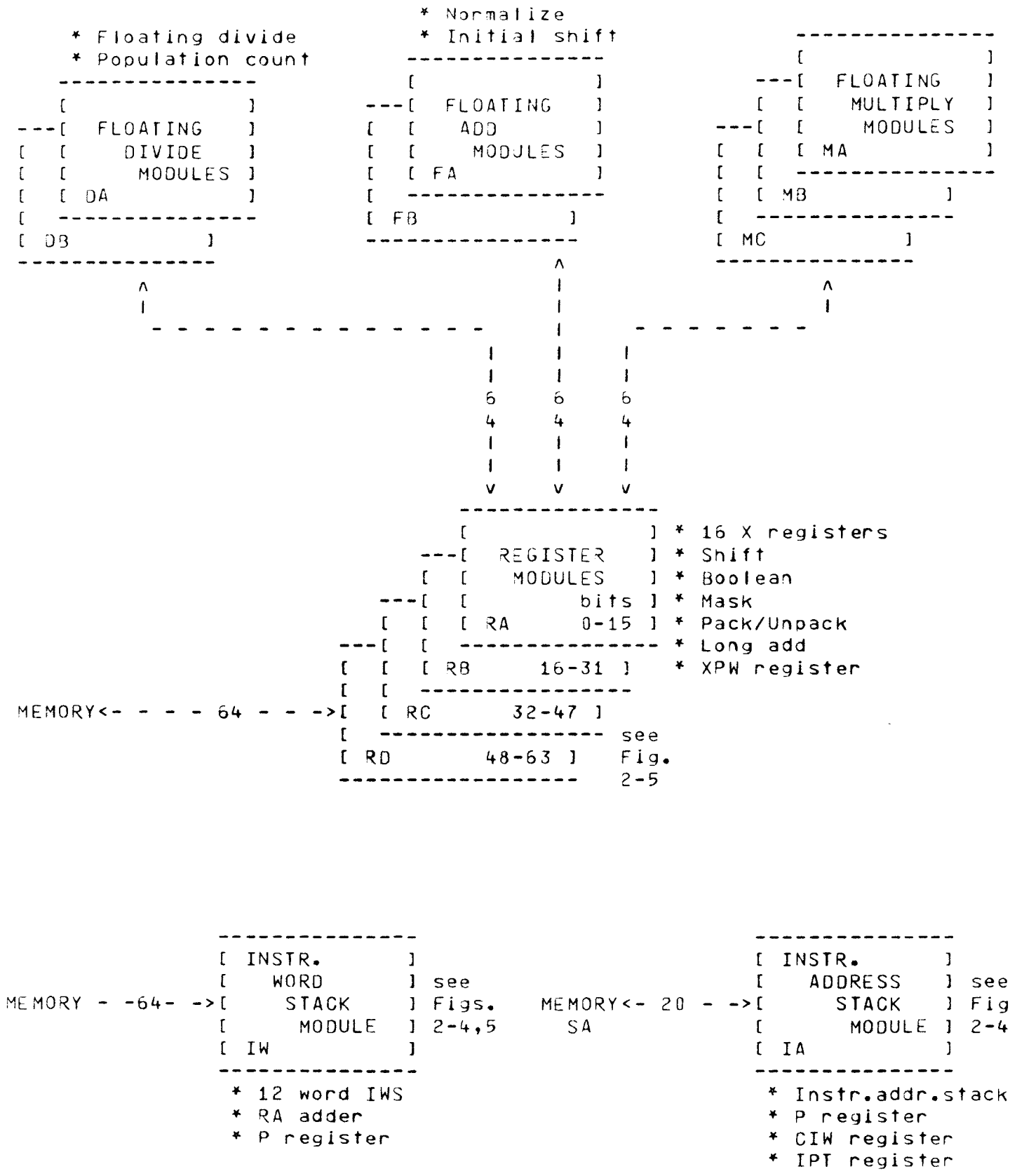


Fig. 2-2 8000 PROCESSOR

Instruction Word Stack & Instruction Address Stack Modules (figs.2-3, 2-4)

The instruction word stack (IWS) contains twelve 64-bit registers which hold program instruction words for execution. The instruction stack information is essentially a moving window in the program code. Each new word entered in the IWS is entered from memory two words ahead of the program instruction word currently being executed, and at the same time the oldest previously executed instruction word in the stack is discarded. The IWS allows the program to branch back to previously executed instructions still in the IWS without referencing memory.

It may be necessary in program code to occasionally complete a 64-bit instruction word with one parcel pass (0033) instructions. This must be done to avoid starting a two parcel instruction in the fourth parcel of an instruction word. One parcel pass instructions are also used to pad out an instruction word so that the next instruction will be the first parcel of an instruction word (this is necessary for branch entry points because a branch instruction destination address must begin with a new word).

Program instruction words in memory and the IWS are divided into four 16-bit fields called parcels. An 8000 instruction may occupy either 1 or 2 parcels. 1-parcel instructions consist of a 4, 6, or 8 bit instruction codes and two or more designators (i, j, k). In 2-parcel instructions, the k designator is expanded into a 20-bit operand K (see page 1-0 for information on instruction word formats).

16	16	16	16
32		32	
16	32		16
16	16	32	
32		16	16

Instruction Combinations in Memory and IWS

64-bit instruction words are read up one at a time from the instruction word stack IWS into a 64-bit current instruction word register CIW. From the CIW each instruction word is gated one parcel at a time to the 16-bit instruction parcel translator IPT where each parcel is interpreted for execution. The IPT controls all of the data transmission paths between the sixteen operating registers and the arithmetic units contained in the four register modules and seven functional unit modules (figure 2-2).

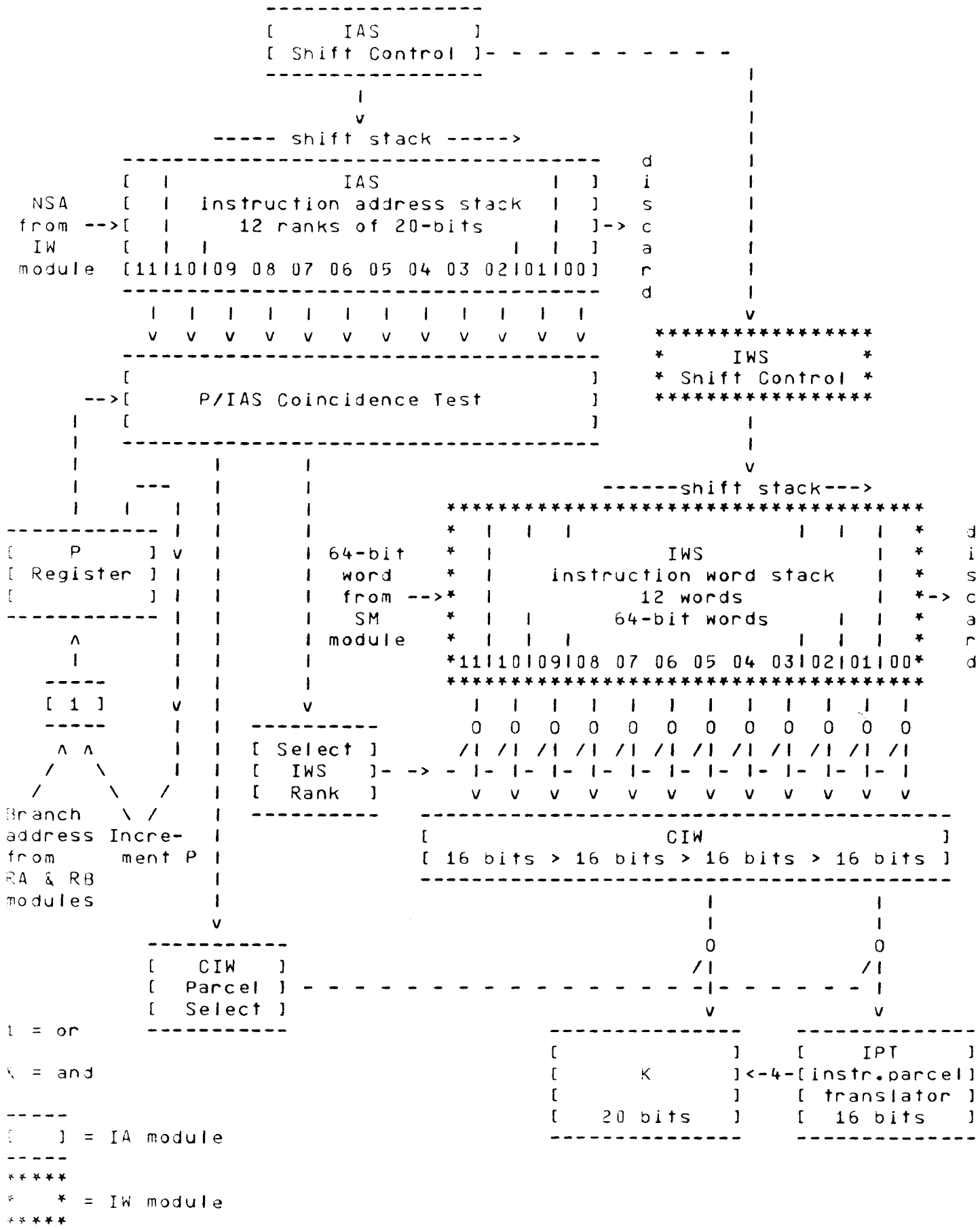


Fig. 2-3 IA and IW MODULES - IWS and CIW Control

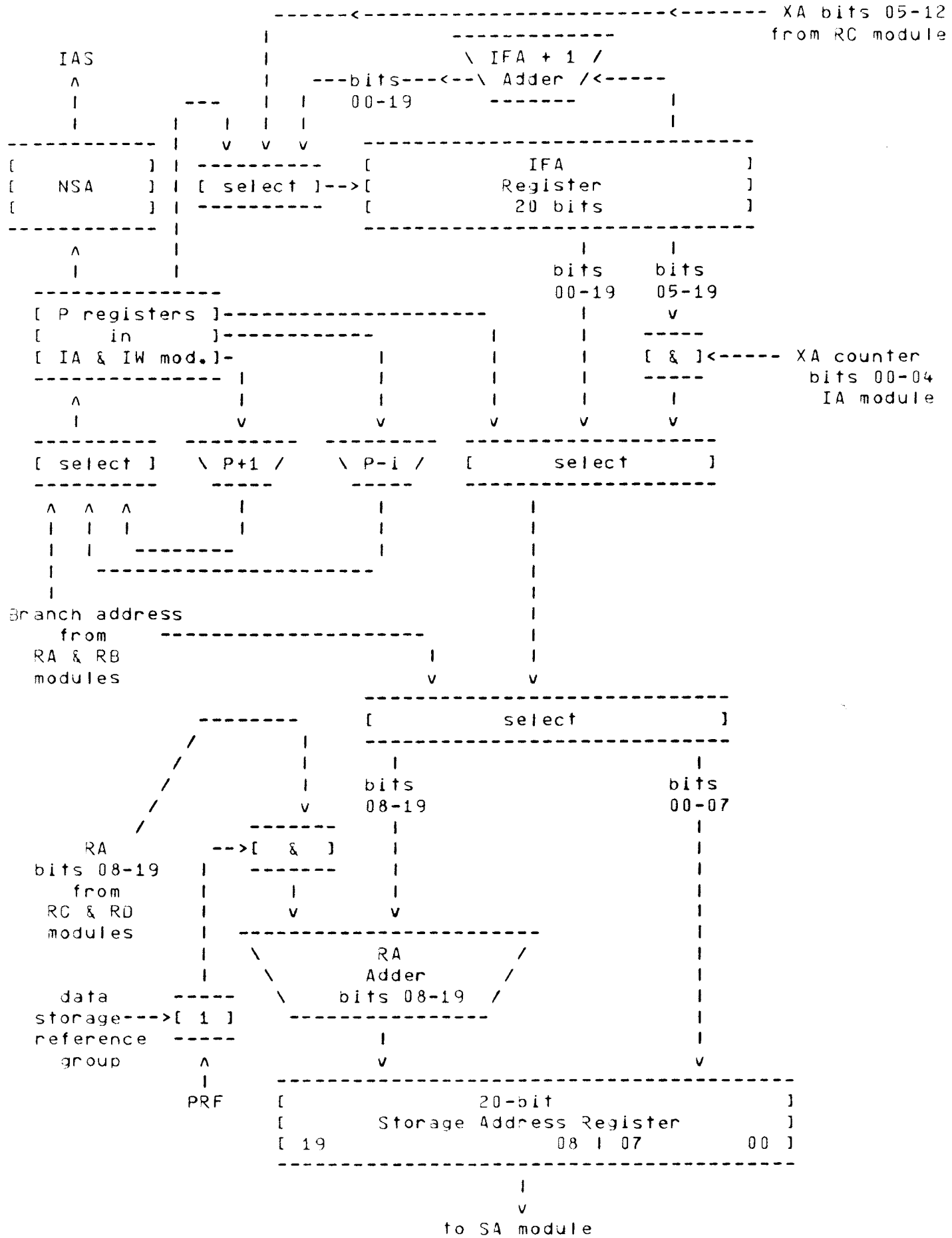


Fig. 2-4 IW MODULE - Storage Address Components

B

Register Modules (figure 2-5)

X registers

16 X registers are the operating registers for each 8000 processor. They are individually designated in this manual by di-bit symbols X00 thru X33. These registers are each 64 bits in length and serve as operand source and destination registers, operand address registers, and indexing registers. Each register is a clear/enter type register with gated clock pulse control. Data will remain in an X register until a control condition generated in the X register access control unit specifically gates a clock pulse to clear the data and enter new data. At most one X register can be cleared and entered with new data at the end of any given clock period.

Communication between the X registers and the arithmetic networks involves a substantial merging of 64-bit data paths and distribution of 64-bit data paths. Almost every arithmetic network has at least one data path to the X registers and one data path from the X registers. The floating point modules have multiple 64-bit paths. The merging and distribution functions are performed in 64-bit static networks preceeding and following the X registers.

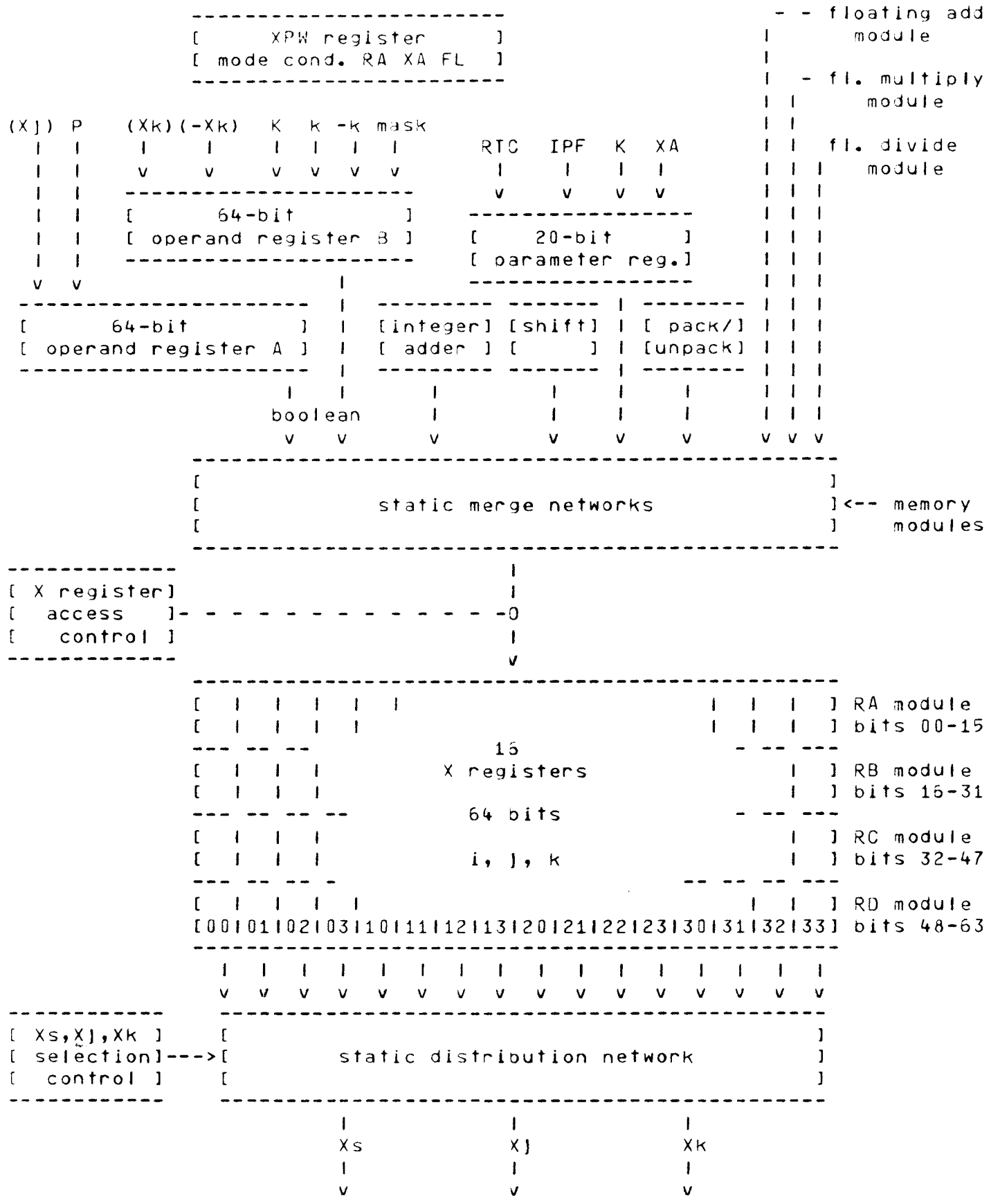


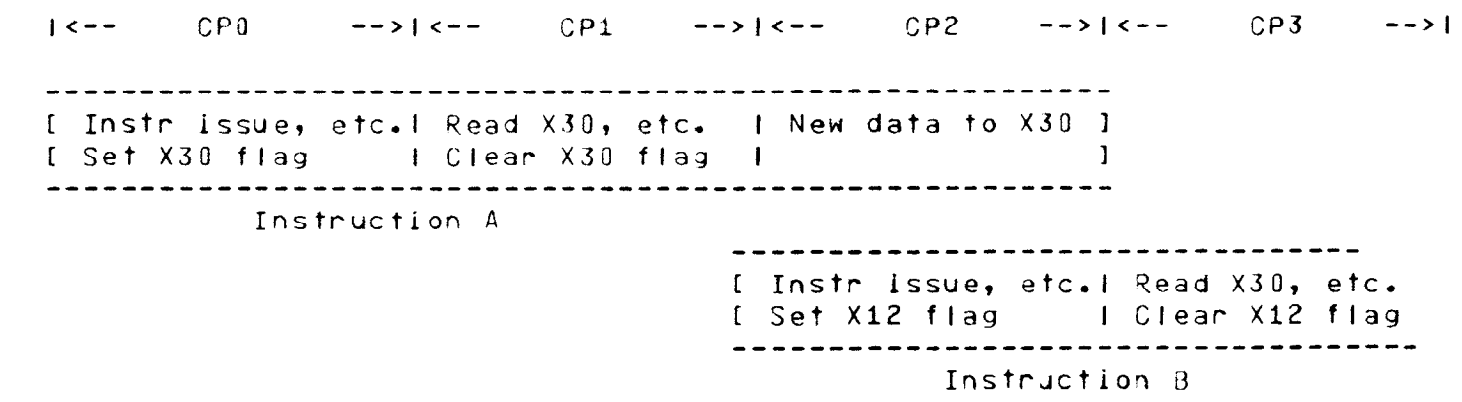
Fig. 2-5 REGISTER MODULES RA, RB, RC, RD

X RESERVATION FLAGS

There is a reservation flag for each of the sixteen X registers in an 8000 processor. When set, the flags remain set until specifically cleared. Set and clear conditions can never both exist in the same clock period. all X reservation flags are forced clear on dead start.

When the instruction parcel translator (IPT) issues an instruction parcel which designates an X register as the destination register, the reservation flag for that register is set. This flag prevents subsequent instructions from reading the contents of the X register until new data has been transmitted to the register. The contents of an X register is always read one clock period after instruction issue, therefore the reservation flag is cleared 1 clock period before new data is transmitted to the register to allow subsequent instructions to read the new data as soon as it is available.

Example:



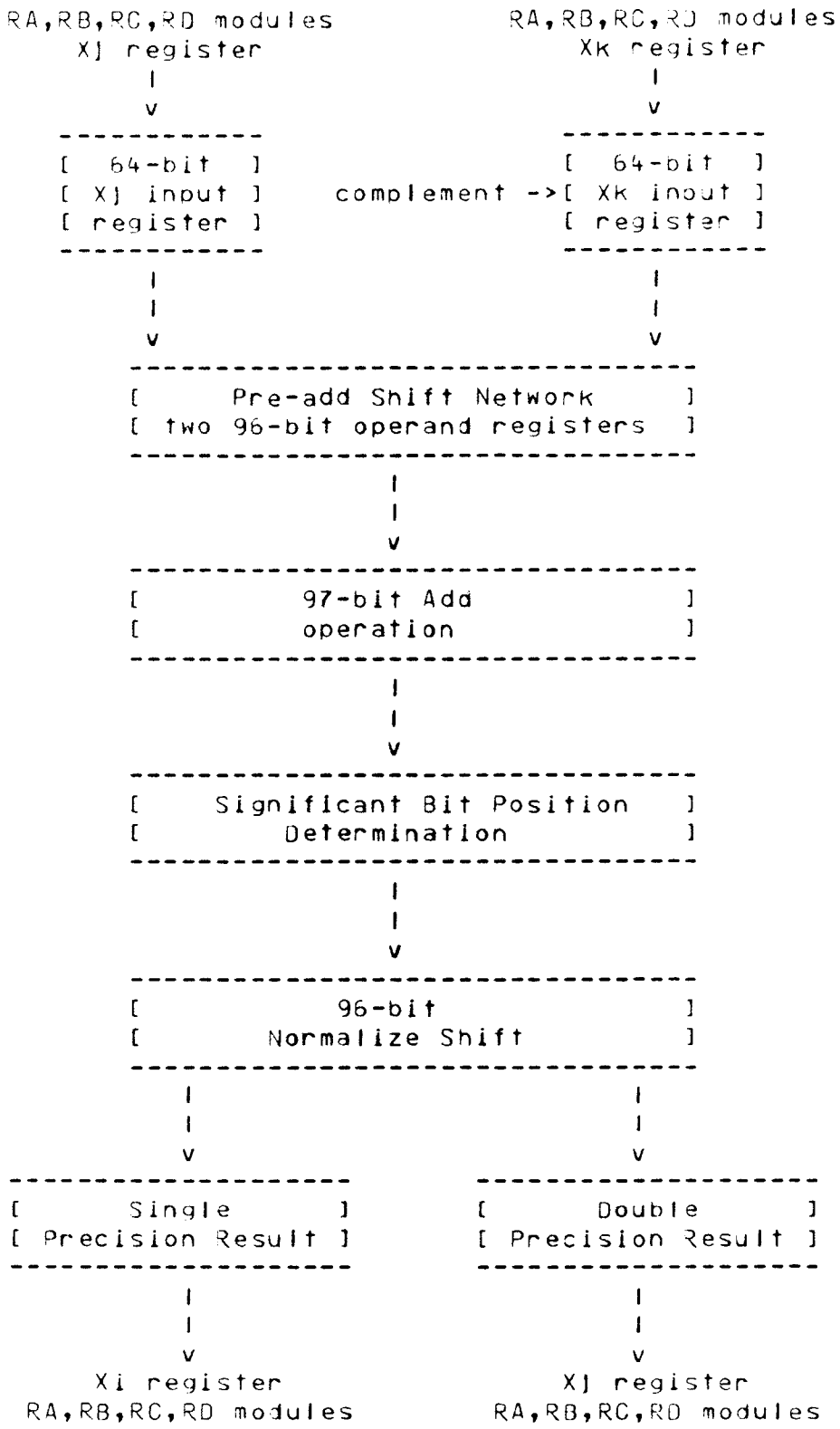


Fig. 2-6 FLOATING ADD MODULES FA, FB

A

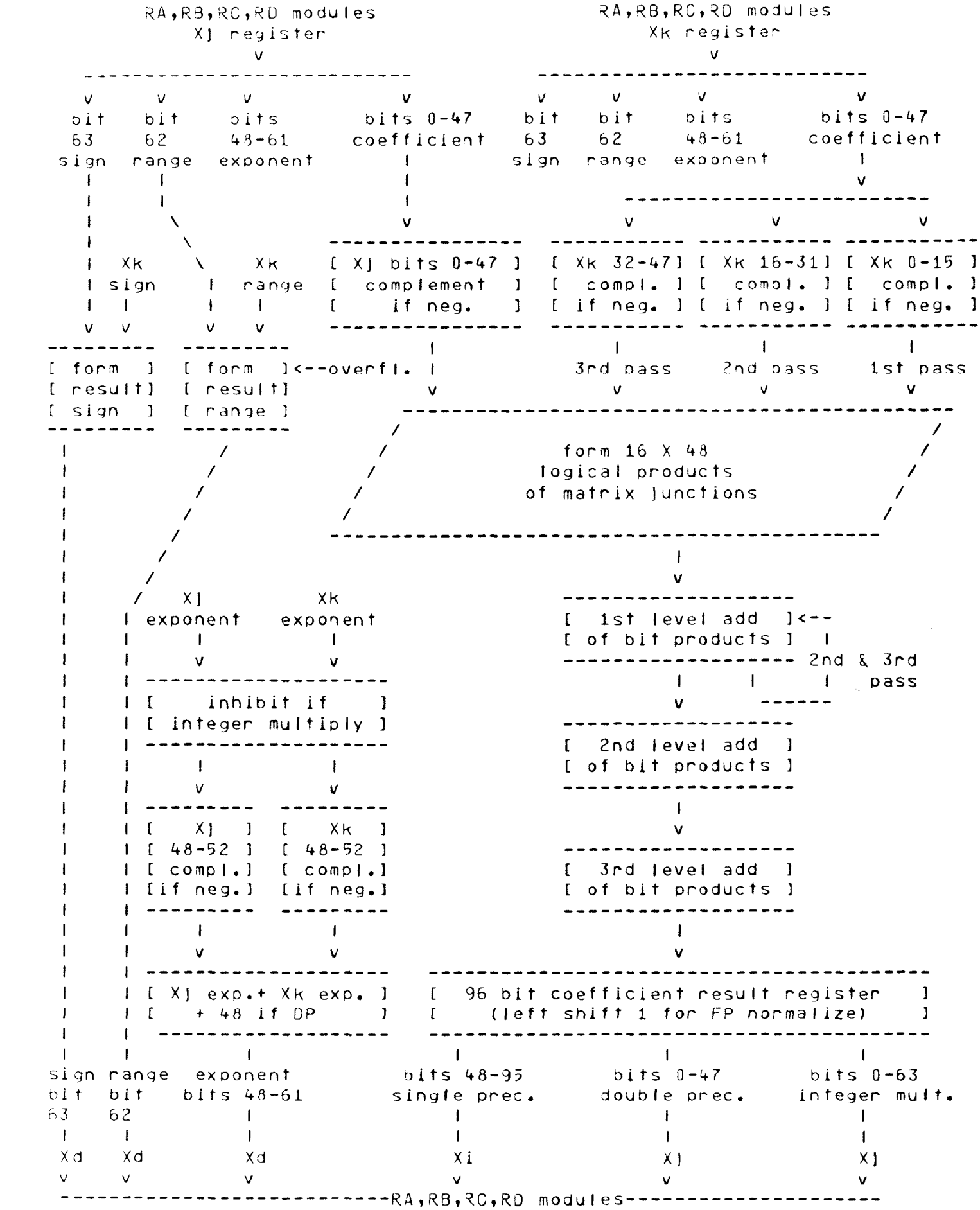


Fig. 2-7 FLOATING MULTIPLY MODULES MA, MB, MC

8600 MEMORY

The overall organization of the 8600 memory is shown in figure 2-10 on the following page. Storage addresses arrive at the storage address stack (SAS) from the IW module. The storage access control (SAC) unit determines the priority of storage access requests when two requests occur simultaneously. SAC also controls the entry of addresses into the storage address stack (SAS). When the SAS data backs up because of memory conflicts, the SAC stops instruction issue until the conflicts have been resolved.

The storage word stack (SWS) is a buffer area for 64-bit data words which are to be written into memory.

The 64 memory bank modules provide a linear selection type core memory with a total capacity of 256K words of 64-bit length (K = 1024). Each 64-bit word is addressed separately. The memory is arranged in 64 banks (one bank per memory bank module) of 4K words each. Each bank is independent of the other 63 banks.

The maximum data transfer rate between memory as a unit (64 banks) and other parts of the system is one word each clock period. Each memory bank has a nine?? clock period access time from arrival of the storage address to readout of the 64-bit word. The total read/write cycle time for a memory bank is 32?? clock periods. In random addressing of memory by all four processors for program data, instructions, and input/output channel data, an average rate of 10 to 15??? memory banks in operation at one time is anticipated.

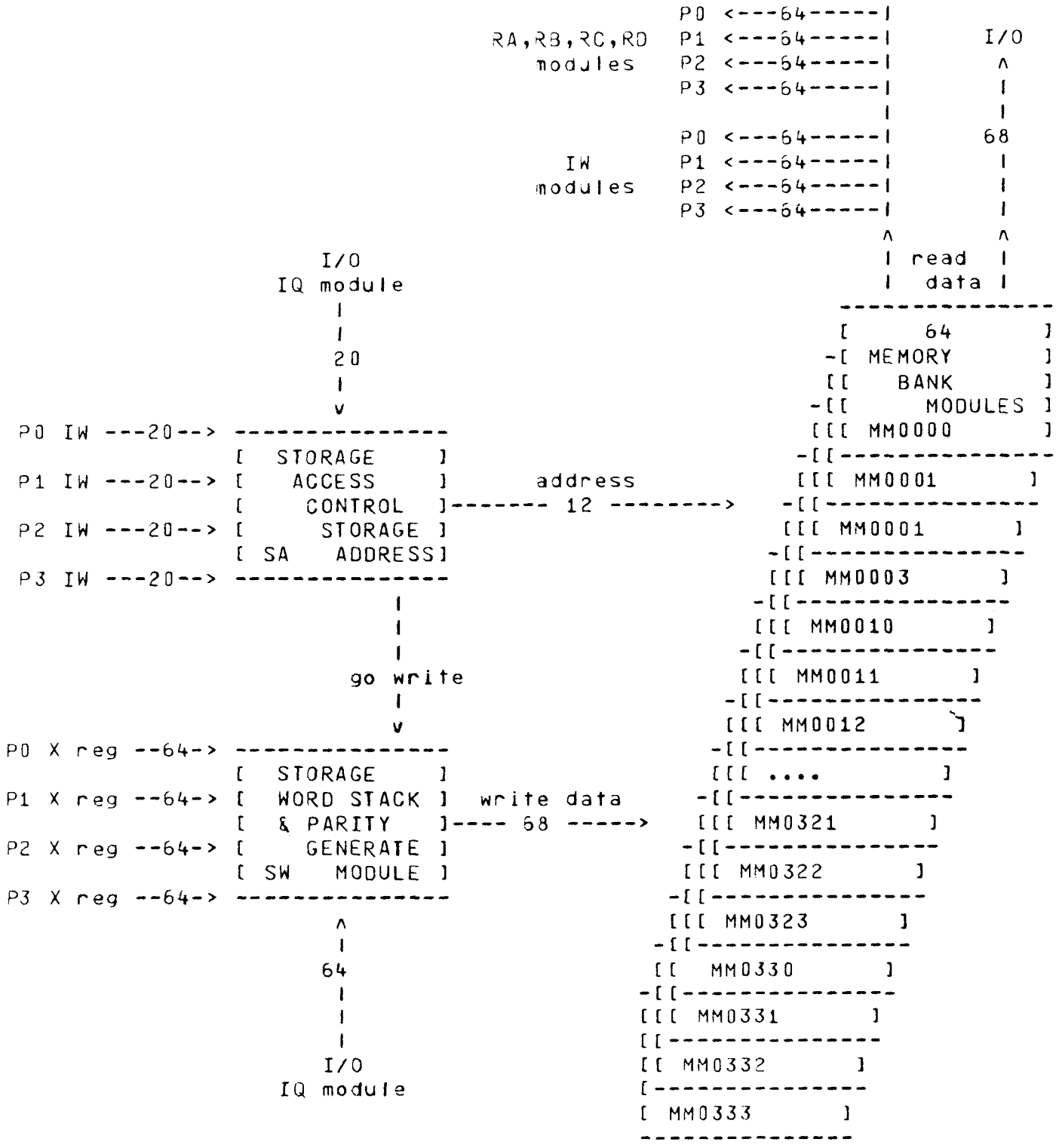


Fig. 2-10 8600 MEMORY

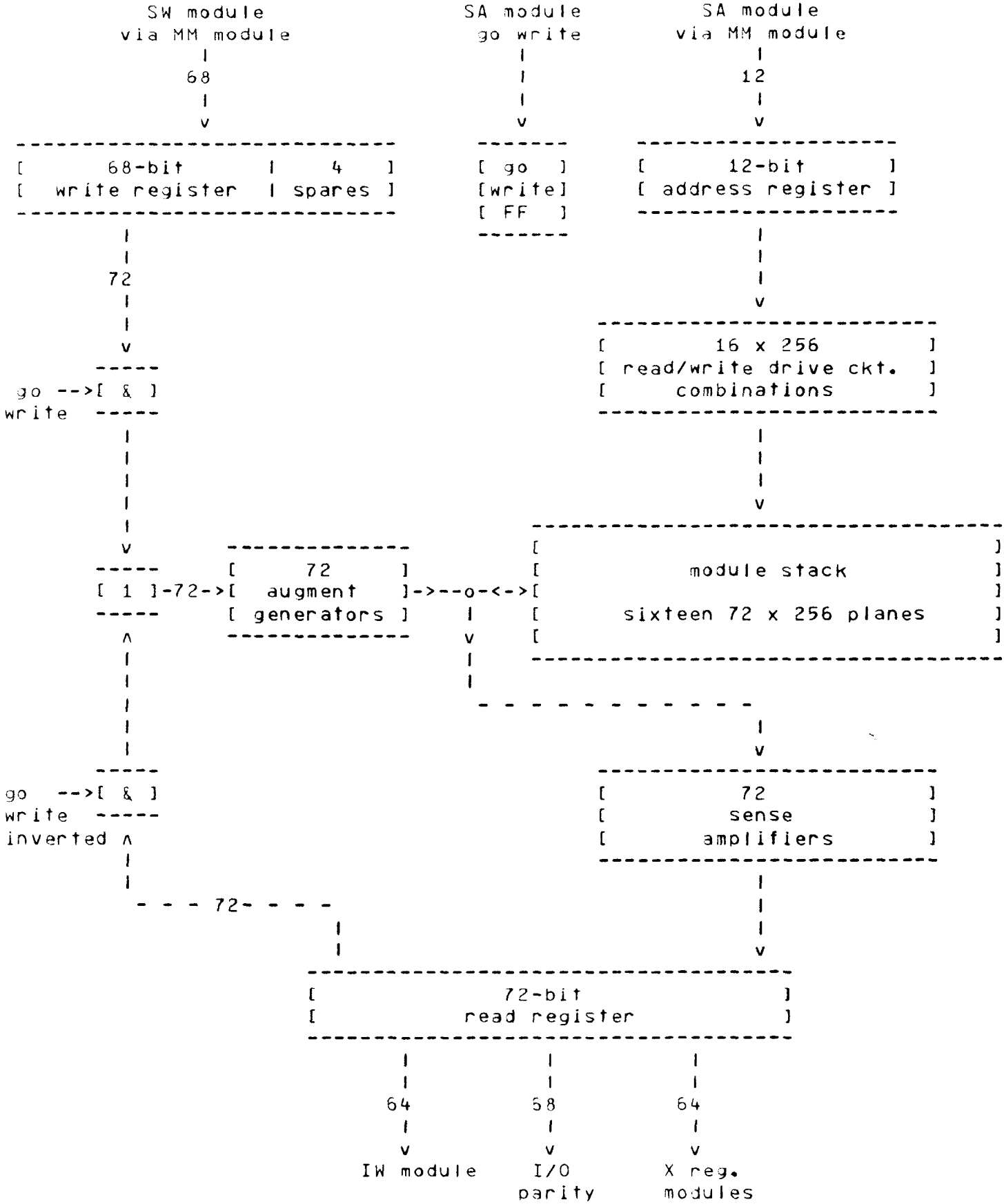


Fig. 2-12 MEMORY BANK MODULE MM

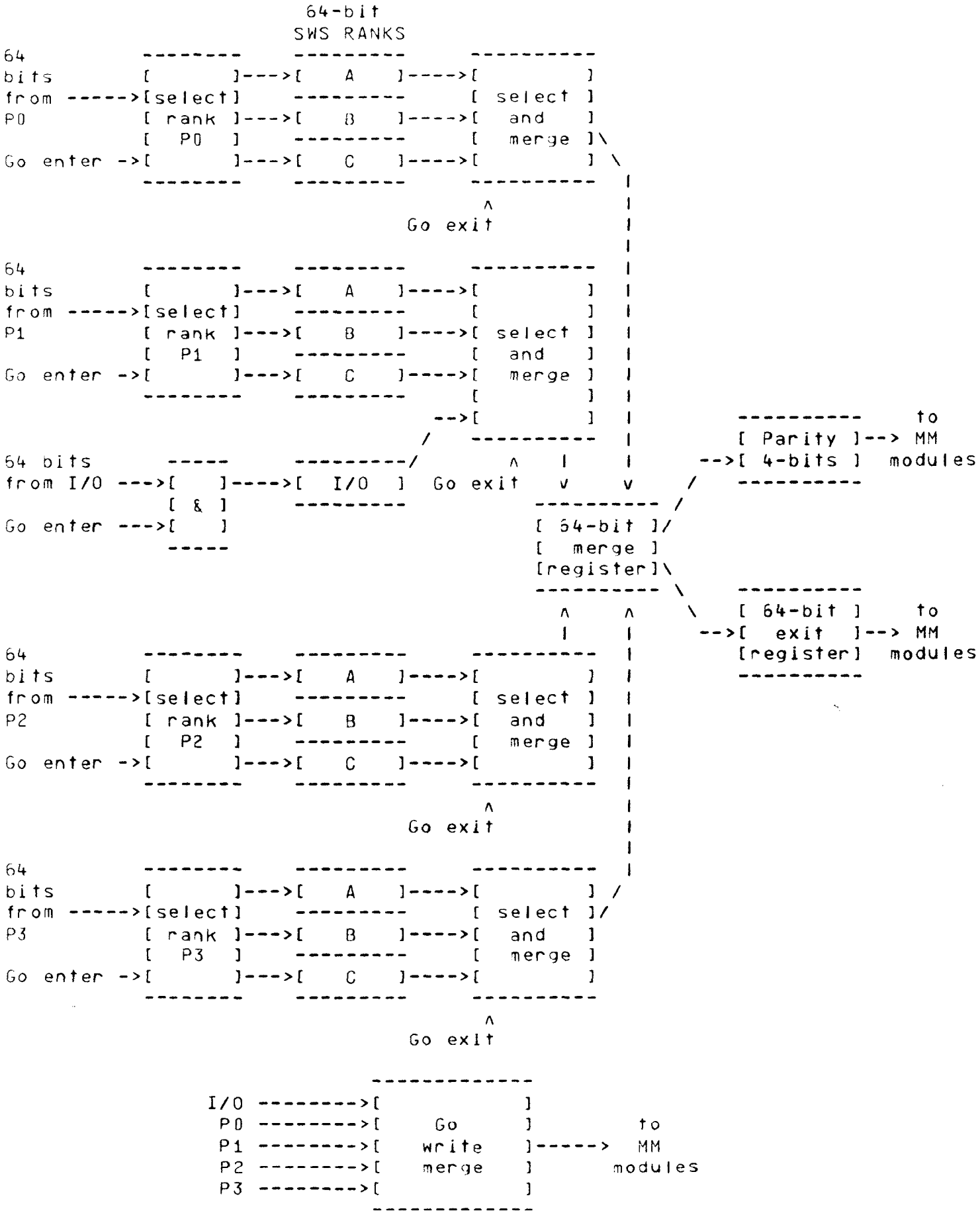


Fig. 2-13 STORAGE WORD STACK MODULE SW

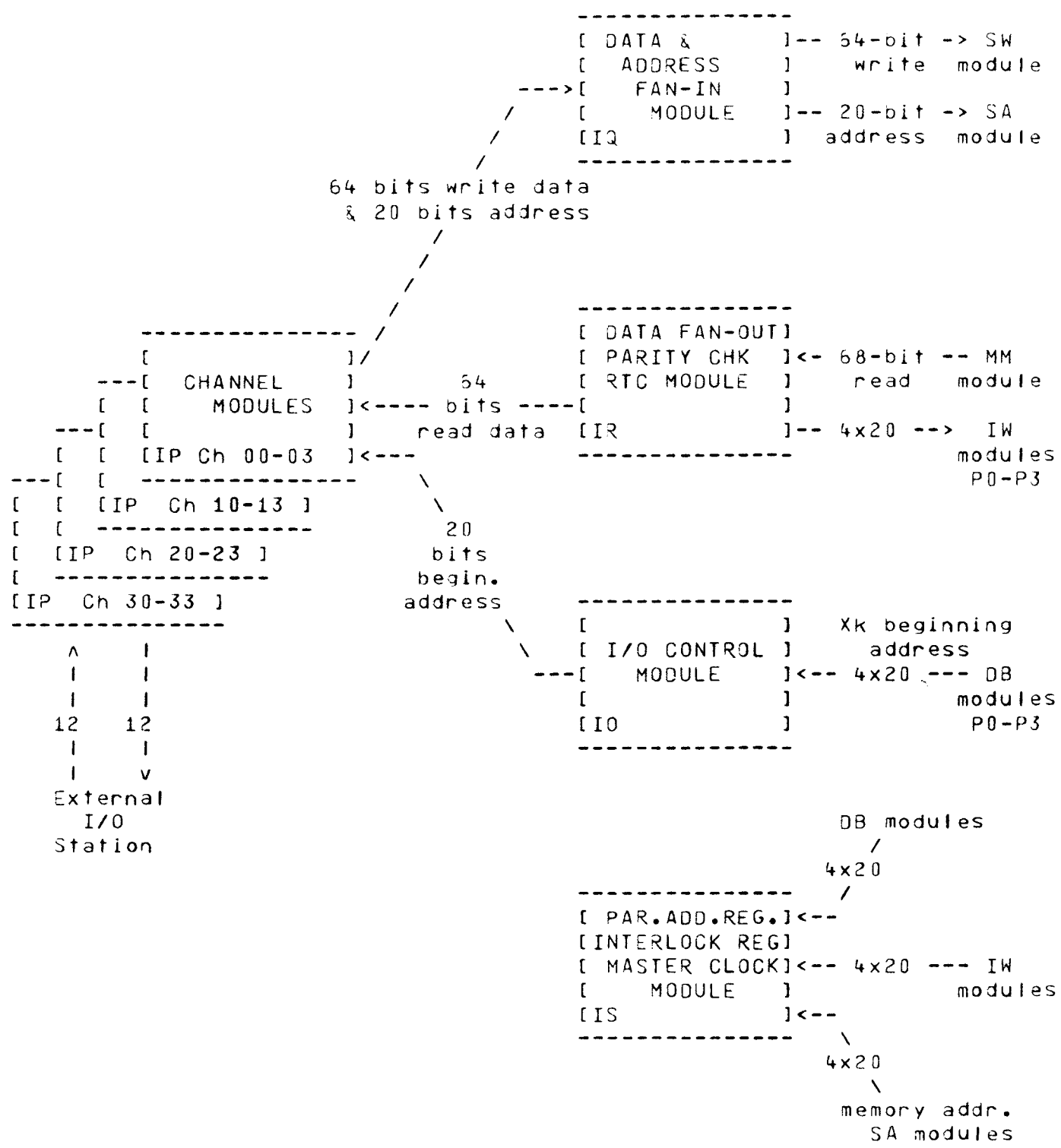


Fig. 2-14 8600 I/O SECTION

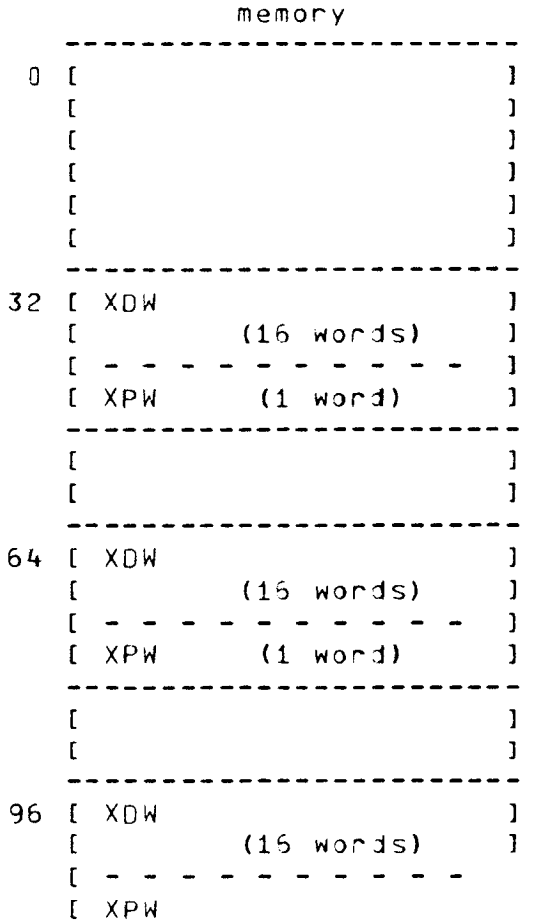
EXCHANGE PACKAGES

An exchange package consists of sixteen 64-bit data words (XDW) and one 64-bit exchange parameter word XPW. The RA and FL fields in the exchange parameter word specify storage increments of 256 word units. XA specifies exchange package location with the lowest order 5 bits removed.

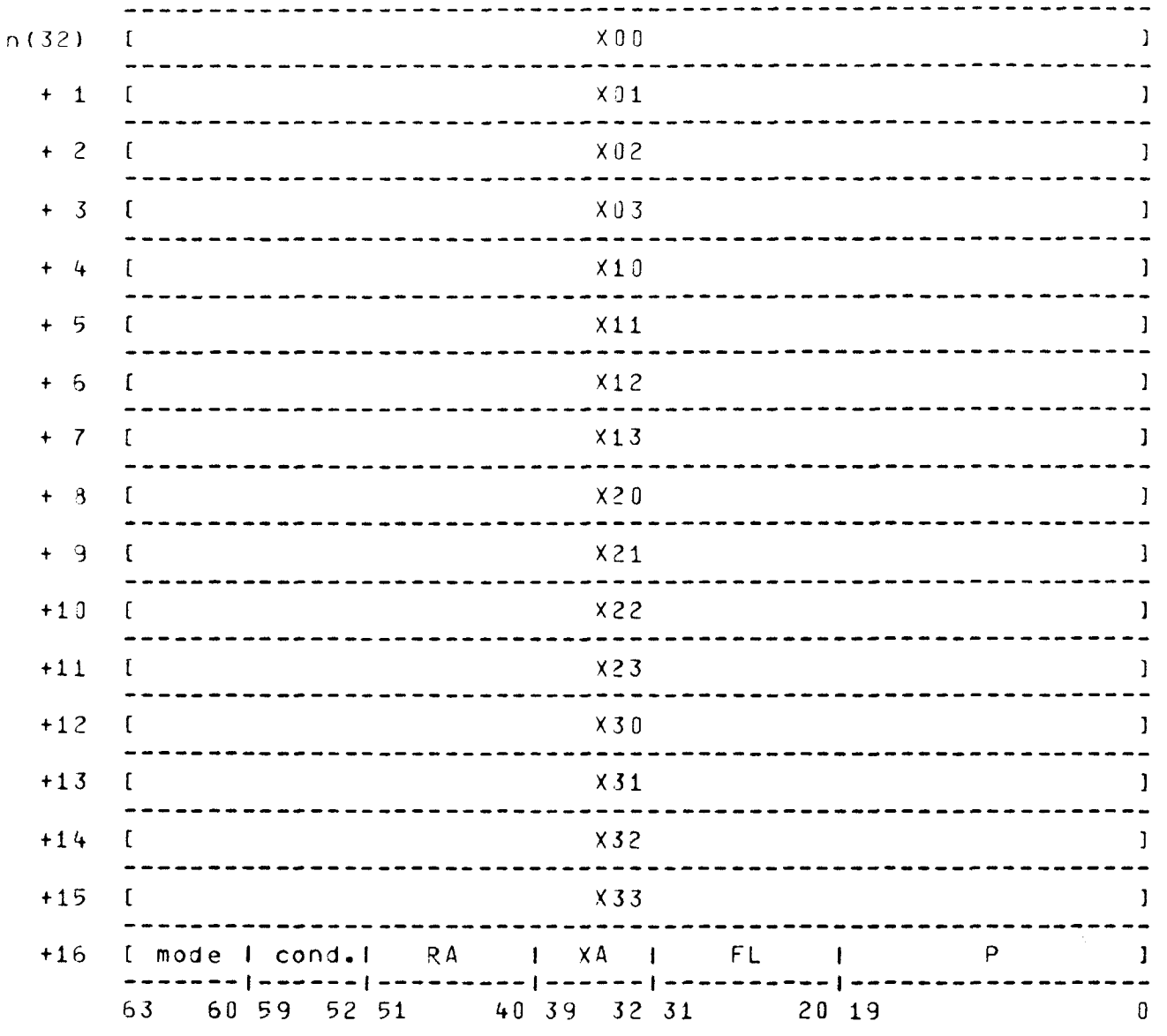
mode	condition	RA	XA	FL	P
4	8	12	8	12	20

Exchange Parameter Word

The exchange package resides in low storage addresses at address multiples of 32. The X register data words XDW appear first in memory followed by the exchange parameter word XPW. The 15 locations following an exchange package can be used for system functions related to the exchange package.



An exchange sequence moves the exchange parameter word first, followed by the X0 register data and the other registers in order. The exchange parameter word (XPW) for the arriving exchange package goes into the X0 register for one clock period, and later moves into a holding register in the register modules (XPW register).



MODE FLAGS

bit	
63	MTF - monitor flag allows I/O, prevents interrupt
62	PRF - program reference flag adds RA to program address
61	IPF - interlock flag allows access to interlock register
60	OVF - FP interrupt interrupts on FP overflow/indefinite

CONDITION FLAGS

bit	
59	Object program call
58	I/O channel request
57	Time interval
56	System call
55	Data field limit
54	Program field limit
53	Program error exit
52	Overflow / indefinite

EXCHANGE PACKAGE

I/O CHANNEL REQUEST

A channel request mechanism is common to all I/O channels. This mechanism is activated by an input RF. The channel number is encoded and presented to all processors over a 4-bit channel request path.

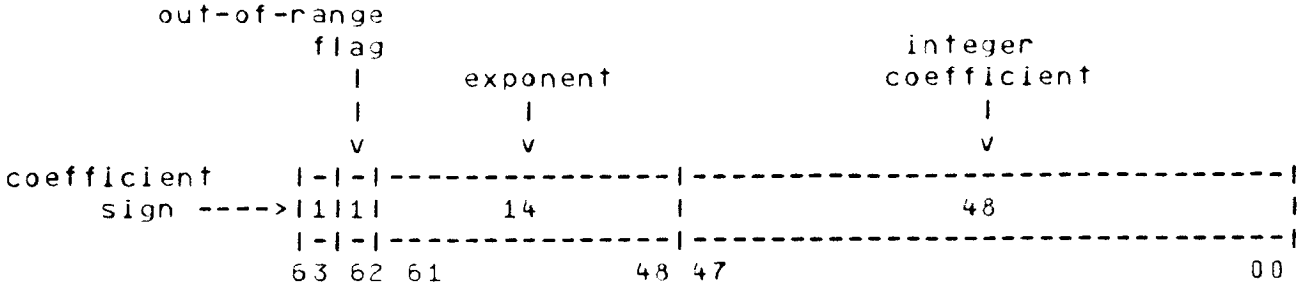
The channel request mechanism scans the processors for a processor with no monitor flag. When one is found, the channel request flag is set in the processor XPW register.

This causes an exchange exit to a monitor program which can read the object program parameter word and determine the cause of the exit. The monitor program can then read the 4-bit channel number from the channel request path. The reading process advances the channel request mechanism to the next channel request, if any.

The monitor program uses the 4-bit channel number for a table lookup to determine the mode of the requesting channel. If the request can be satisfied by the monitor program, the interrupted object program is resumed. If a new program must be initiated, the monitor program updates the running time for the terminated program and exchanges to a new XA.

FLOATING POINT ARITHMETIC

Floating point arithmetic calculations are peformed in the 8000 processors using a packed 64-bit format for number representation. This format represents a signed binary integer coefficient times two with a signed binary integer exponent. The coefficient field contains 49 bits and the exponent field 14 bits. The remaining bit is used to indicate an out-of-range condition.



Floating point format

The coefficient field in the floating point format is not contiguous. The sign of the coefficient occupies the highest order bit position in the 64-bit word. The remainder of the coefficient resides in the lowest order 48 bit positions. If the exponent field and the error flag are replaced with copies of the sign bit (as in the unpack instruction 0110) the resulting format is the normal integer representation.

The exponent field in the floating point format occupies the bit positions 2**48 through 2**61. This field represents exponent values ranging from 2**(-13) to 2**(+13). The bits in the exponent field in the packed floating point format consist of the bits in the integer exponent (in ones complement form) with the highest order bit complemented. The complementing of the highest order exponent bit results in a format which represents the floating point numbers in such a way that their increasing values are also monotonically increasing when viewed as 64-bit integers. As a result of this property, comparison tests of two floating point quantities can be done in the integer adder rather than in the slower floating point adder. Floating point quantities with negative coefficients are packed with the exponent field complemented in order to preserve the above relationship for negative numbers.

Bit position 2**62 in the floating point format contains the out-of-range flag. This flag is set when the exponent in a floating point calculation exceeds 2**(+13) or if the result is indefinite. This flag is considered set when the values of the two highest order bits in the floating point format disagree. Further floating point operations in which this flag appears set in one of the operands results in aborting the normal sequence and generating a result with the out-of-range flag set.

Floating point calculations which underflow the exponent range are aborted and the result replaced with a word of all zero bits.

8

BINARY ARITHMETIC

Binary arithmetic in the 8000 processors is performed in a modified ones complement additive mode. The sum of two binary numbers in a normal ones complement additive mode is defined by the recursive Boolean expressions below.

```
Let m = number of bit positions in adder
    A(i) = addend bit i
    B(i) = augend bit i
    C(i) = carry into bit position i
    S(i) = sum bit i

Where i = 0,1,2,3,4,...,m-1

Then C(i+1) = A(i).and.B(i).or.B(i).and.C(i).or.C(i).and.A(i)
     C(0) = C(m)
     S(i) = A(i).and..not.B(i).and..not.C(i).or.
            B(i).and..not.C(i).and..not.A(i).or.
            C(i).and..not.A(i).and..not.B(i).or.
            A(i).and.B(i).and.C(i)
```

The modification to the above mode consists of replacing a resulting sum of all one bits with a result of all zero bits. An 8000 processor adder therefore has only one form of zero as a resulting sum.

Subtraction is performed by complementing the subtrahend and adding to the minuend.

Part 3

INSTRUCTION DESCRIPTIONS

8000 INSTRUCTION CODES

0000	Program error exit	0220	Read program (P + K) to Xj
0001	Logical Xj * Xk to Xj	0221	Transmit P + K to Xj
0002	Logical Xj + Xk to Xj	0222	Transmit K to Xj
0003	Logical Xj - Xk to Xj	0223	Transmit XA to Xj
0010	Copy Xk to Xj	0230	Set IP flags from Xk (IPF)
0011	Complement Xk to Xj	0231	Clear IP flags from Xk (IPF)
0012	Shift Xj left by Xk	0232	Read IP flags to Xj
0013	Shift Xj right by Xk	0233	Read internal clock to Xj
0020	Floating DP Xj + Xk to Xj	0300	Jump to P + K
0021	Floating DP Xj - Xk to Xj	0301	Call subroutine at P + K
0022	Floating Xj / Xk to Xj	0302	Jump to P + K if Xj in range
0023	Population Xk to Xj	0303	Jump to P + K if Xj error
0030	Floating DP Xj * Xk to Xj	0310	Jump to P + K if Xj is zero
0031	Integer Xj * Xk to Xj	0311	Jump to P + K if Xj not zero
0032	Program error exit	0312	Jump to P + K if Xj positive
0033	Pass	0313	Jump to P + K if Xj negative
0100	Transmit k to Xj	0320	Call subroutine at K
0101	Transmit -k to Xj	0321	Call subroutine at Xk
0102	Integer Xj + k to Xj	0322	Call libr. routine at K [cl PRF]
0103	Integer Xj - k to Xj	0323	Call libr. routine at Xk [cl PRF]
0110	Unpack coefficient Xj to Xk	0330	Subroutine exit to Xj + k
0111	Unpack exponent Xj to Xk	0331	Lib. exit to Xj + k [set/cl PRF]
0112	Pack Xk * 2 ** Xj to Xk	0332	Jump to K
0113	Integer 0 - Xk to Xj	0333	Exchange exit
0120	Begin system call [MTF]	100X	Save lower Xj for n bits
0121	End system call [MTF]	101X	Blank lower Xj for n bits
0122	Channel Xj to (Xk) [MTF]	102X	Left shift Xj by n bits
0123	Channel Xj from (Xk) [MTF]	103X	Right shift Xj by n bits
0130	Channel request to Xj [MTF]	11XX	Integer Xj + K to Xi
0131	Load XA from Xk [MTF]	12XX	Integer Xj + Xk to Xi
0132	Program error exit	13XX	Integer Xj - Xk to Xi
0133	Program error exit		
0200	Store data (K) from Xj	20XX	Floating Xj + Xk to Xi
0201	Store data (Xk) from Xj	21XX	Floating Xj - Xk to Xi
0202	Read/store data (K) and Xj	22XX	Floating Xj * Xk to Xi
0203	Read/store data (Xk) and Xj	23XX	Jump to P - i if Xj < Xk
0210	Read data (K) to Xj	30XX	Read data (Xj + K) to Xi
0211	Read data (Xk) to Xj	31XX	Read data (Xj + Xk) to Xi
0212	Read program (K) to Xj	32XX	Store data (Xj + K) from Xi
0213	Read program (Xk) to Xj	33XX	Store data (Xj + Xk) from Xi

C

NOTES

1. The following pages of instruction descriptions are numbered with the instruction codes shown in the box in the upper right corner of each page. The instruction codes are shown in quaternary or dibit notation and parenthetically in hexidecimal notation.

[0331]	[103X]	[23XX]
[(3D)]	[(4C)]	[(Bx)]
8-bit Instruction code	6-bit Instruction code	4-bit Instruction code

2. This section makes extensive use of abbreviations and special terms which are listed in the index in Appendix A together with a reference to the page where each term is defined.

   Parenthesis are used to indicate the contents of a register, e.g., (XJ) indicates the contents of the X register specified by the J designator.

B

0000xxxx	Program error exit	[0000] [ ] [(00)]

This instruction format is treated as an error condition and, if executed, will set the program error exit flag in the exchange parameter word (XPW). This condition flag will then cause an exchange jump to address (XA). In this case all instructions which have issued prior to this instruction will be run to completion. Any instructions following this instruction in the current instruction word will not be executed. When all operands have arrived at the operating registers as a result of the previously issued instructions, an exchange jump will occur to the exchange package which is designated by (XA).

The j and k designators in this instruction are ignored. The program address stored in the exchange package on the terminating exchange jump is advanced one count from the address of the current instruction word. This is true no matter which parcel of the current instruction word contains the program error exit instruction.

This instruction format is not intended for use in normal program code. The program error exit flag is set in the exchange parameter word (XPW) to indicate that the program may be in range but is not executing valid program code. This could occur when an incorrectly coded program jumps into an unused area of the memory field or into a data field.

0001jjkk	Logical product of (Xj) and (Xk) to Xj	[0001] [(01)]

This instruction causes operands to be read from the Xj and Xk registers, forms a bit-bit logical product, and enters the result in the Xj register. Each of the bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in the result entered in the Xj register. A sample computation is listed below in di-bit notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
Sample operands:  (Xj) = 0101
 (lower 4 bits)   (Xk) = 1100
    (binary)             ----
                  (Xj) = 0100
```

This instruction is intended for extracting portions of a 64-bit word during data processing as distinguished from numerical computation. Together with other boolean and shift instructions it may be used to manipulate alphanumeric or other coded data not related to the 64-bit machine word length.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Read (Xk) to operand register B
Clear Xj reservation flag

CP2 Transmit logical product of (A) and (B) to Xj

NOTES

1. If the f and k designators have the same value this instruction will read a 64-bit word from the designated X register and then write the same information back into that X register. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

0002jjkk	Logical sum of (Xj) plus (Xk) to Xj	[0002] [ ] [(02)]

This instruction causes operands to be read from the Xj and Xk registers, forms a bit-bit logical sum, and enters the result in the Xj register. Each of the 64 bits in (Xj) is acted upon by the corresponding bit of (Xk) to form a single bit in the result entered in the Xj register. A sample computation is listed below in di-bit notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

```
Sample operands:  (Xj) = 1010
 (lower 4 bits)   (Xk) = 1100
     (binary)           ----
                  (Xj) = 1110
```

This instruction is intended for merging portions of a 64-bit word into a composite word during data processing as distinguished from numerical computation. Together with the other boolean and shift instructions it may be used to manipulate alphanumeric or other coded data not related to the 64-bit machine word length.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Read (Xk) to operand register B
Clear Xj reservation flag

CP2 Transmit logical sum of (A) and (B) to Xj

NOTES

1. If the j and k designators have the same value this instruction will read a 64-bit word from the designated X register and then write the same information back into that X register. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

0003jjkk	Logical difference of (Xj) minus (Xk) to Xj	[0003] [ ] [(03)]

This instruction reads operands from the Xj and Xk registers, forms the bit-by-bit logical difference of (Xj) minus (Xk), and enters the resulting 64-bit word in the Xj register.

Sample operands:	(Xj) = 1010
(lower 4 bits)	(Xk) = 1100
(binary)	----------
	(Xj) = 0110

This instruction is intended for comparing bit patterns or for complementing bit patterns during data processing as distinguished from numerical computation. Together with the other boolean and shift instructions it may be used to manipulate alphanumeric or other coded data not related to the 64-bit machine word length.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Read (Xk) to operand register B
Clear Xj reservation flag

CP2 Transmit logical difference of (A) and (B) to Xj

NOTES

1. If the j and k designators have the same value in this instruction, a logical difference is formed between two identical quantities. The result will be a word of all 0's written into register Xj. The timing for this case is the same as the timing for the general case.

```
------------                                                  --------
| 0010|jkk |  Copy (Xk) to Xj                                 [ 0010 ]
------------                                                  [      ]
                                                              [ (04) ]
                                                              --------
```

This instruction reads a 64-bit word from the Xk register and enters the word in the Xj register.

This instruction is intended for moving data from one X register to another X register as rapidly as possible. No logical function is performed on the data.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xk) to operand register B
Clear Xj reservation flag

CP2 Transmit (B) to Xj

NOTES

1. If the j and k designators have the same value this instruction will read a 64-bit word from the designated X register and then write the same information back into that X register. The timing for this case will be the same as the timing for the general case, and no special conflicts will occur.

0011jjkk	Copy complement of (Xk) to Xj	[0011] [ ] [(05)]

This instruction reads a 64-bit word from the Xk register, complements the the word, and enters the result in the Xj register.

This instruction is also useful in data processing for inverting an entire 64-bit field. The j and k designators may frequently have the same value because the result may often be returned to the same X register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xk) complement to operand register B
Clear Xj reservation flag

CP2 Transmit (B) to Xj

NOTES

1. The j and k designators may frequently have the same value in this instruction. In this case, the quantity read from the designated X register is complemented and returned to the same X register. The timing is the same as for the general case.

```
------------                                                    --------
| 0012jjkk |  Shift (Xj) left by (Xk)                          [ 0012 ]
------------                                                    [      ]
                                                                [ (06) ]
                                                                --------
```

This instruction reads a 64-bit operand from the Xj register, shifts the operand left or right as specified by (Xk), and enters the resulting operand back into the Xj register. If (Xk) is positive, the data is shifted left circularly the number of bit positions designated by (Xk). If (Xk) is negative the data is shifted right (end off) with sign extension the number of bit positions designated by the magnitude of (Xk). Sample shift operations are listed below in binary notation.

```
  Sample (64 bits):       (Xj) operand  =  10110000......00000000
        (Xk) = 0..0100    (Xj) result   =  00000000......00001011


  Sample (64 bits):       (Xj) operand  =  01100000......00001000
        (Xk) = 1..1101    (Xj) result   =  00011000......00000010


  Sample (64 bits):       (Xj) operand  =  11000000......00100010
        (Xk) = 1..1100    (Xj) result   =  11111000......00000100
```

This instruction is for use in data processing where the shift count (Xk) is derived by program computation. It is also used for generating a truncated integer from the coefficient of a floating point number when the exponent has been unpacked into an X register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

```
CP0     Instruction issues from IPT
        Set Xj reservation flag

CP1     Read (Xj) to operand register A
        Read (Xk) to operand register B

CP2     Begin operand shift
        Clear Xj reservation flag

CP3     Complete operand shift
        Transmit result to Xj
```

NOTES

1. The maximum shift count which may be specified by (Xk) is 63 decimal.

2. If (Xk) is 0 (either all 1's or all 0's binary) the instruction reads the operand from register Xj and returns it unaltered to register Xj. The timing for this case is the same as for the general case.

```
------------                                                    --------
| 0013JJkk |  Shift (Xj) right by (Xk)                        [ 0013 ]
------------                                                  [      ]
                                                              [ (07) ]
                                                              --------
```

This instruction reads a 64-bit operand from the Xj register, shifts the operand right or left as specified by (Xk), and writes the resulting operand back into the Xj register. If (Xk) is positive, the operand is shifted right (end off) with sign extension the number of bit positions designated by (Xk). If (Xk) is negative, the operand is shifted left circularly the number of bits designated by the magnitude of (Xk). Sample shift operations are listed below in binary notation.

```
  Sample  (64 bits):        (Xj) operand  =  10110000......00000000
          (Xk) = 0..0100    (Xj) result   =  11111011......00000000


  Sample  (64 bits):        (Xj) operand  =  01100000......00001000
          (Xk) = 1..1101    (Xj) result   =  10000000......00100001
```

This instruction is for use in data processing where the shift count (Xk) is derived by program computation.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after instruction issues from IPI.

CP0 Instruction issues from IPI
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Read (Xk) complement to operand register B

CP2 Begin operand shift
Clear Xj reservation flag

CP3 Complete operand shift
Transmit result to Xj

NOTES

1. The maximum shift count which may be specified by (Xk) is 63 decimal.

0020 jjkk	Floating double precision sum of (Xj) plus (Xk) to Xj	[0020] [ ] [(08)]

This instruction forms the floating double precision sum of two floating point operands read from the Xj and Xk registers, and enters the lower half of the result in the Xj register.

The two operands are not rounded in this operation and may or may not be normalized. They are unpacked from floating point format and the exponents compared. The coefficient with the smaller exponent is shifted down by the difference of the exponents so as to align bits of corresponding significance, and a 97-bit adder forms a double precision 1's complement sum. The exponent of the 48-bit lower half coefficient entered in Xj is 48 decimal less than the exponent of the upper half coefficient.

The 97-bit result is normalized following the add operation. The result coefficient is displaced right one bit or left up to 64 bits and the result exponent is incremented or decremented by the shift count.

Since the double precision sum is normalized, the upper half result is normalized, but the lower half result is not.

A zero result from this instruction is always a positive zero.

This instruction is intended for use in floating point calculations involving double precision or multiple precision. Used together with the single precision FP sum instruction 20XX, this instruction forms a double precision sum in two X registers with no loss of precision.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issue
Xk register is free one clock period after instruction issue
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

EXECUTION TIMING (continued)

CP0	Instruction issues from IPT Transmit J and k designators to register modules Set XJ reservation flag
CP1	Read (XJ) to floating add module FA Read (Xk) to floating add module FA Compare exponents Transmit coefficients to pre-add shift register
CP2	Select smaller exponent
CP3	Shift coefficients for bit alignment Transmit coefficients to 97-bit adder
CP4	Form double precision sum
CP5	Transmit DP sum to bit position network
CP6	Determine significant bit position Transmit result to shift network
CP7	Perform 96-bit normalize shift Clear XJ reservation flag
CP8	Enter lower half of result in XJ

I 0021JJKK I	Floating double precision difference of (XJ) minus (Xk) to XJ	[0021] [ ] [(09)]

This instruction forms the floating double precision difference of two floating point operands read from the XJ and Xk registers, and enters the lower half of the (XJ) minus (Xk) result in the XJ register.

The two operands are not rounded in this operation and may or may not be normalized. (XJ) and complemented (Xk) are unpacked from floating point format, the coefficient with the smaller exponent is shifted down by the difference of the exponents, and a 97-bit adder forms a double precision 1's complement sum. The exponent of the 48-bit lower half coefficient entered in XJ is 48 decimal less than the exponent of the upper half coefficient.

The 97-bit result is normalized following the add operation. The result coefficient is desplaced right one bit or left up to 64 bits and the result exponent is incremented or decremented by the shift count.

Since the double precision difference is normalized, the upper half result is normalized, but the lower half result is not.

This instruction is intended for use in floating point calculations involving double precision or multiple precision. Used together with the single precision FP difference instruction 21XX, this instruction forms a 96-bit double precision difference in two X registers with no loss of precision.

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

EXECUTION TIMING (continued)

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to floating add module FA
Read complement of (Xk) to floating add module FA
Compare exponents
Transmit coefficients to pre-add shift register

CP2 Select smaller exponent

CP3 Shift coefficients for bit alignment
Transmit coefficients to 97-bit adder

CP4 Form double precision sum

CP5 Transmit DP sum to bit position network

CP6 Determine significant bit position
Transmit result to shift network

CP7 Perform 96-bit normalize shift
Clear Xj reservation flag

CP8 Enter lower half of result in Xj

0022JJKK Floating divide of (XJ) by (XK) to XJ	[0022] [ ] [(0A)]

This instruction reads operands from the XJ and Xk registers, forms a floating point quotient, and delivers this result to the XJ register. The dividend operand is (XJ) and the divisor operand is (Xk). The remainder from the division process is discarded.

The operands are assumed to be numbers in floating point format. If the divisor operand is not normalized, the out-of-range flag is set.

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free clock periods after issue

EXECUTION TIMING

0023jjkk Population count of (Xk) to Xj	[0023] [ ] [(0B)]

This instruction reads an operand from the Xk register, counts the number of 1 bits in the operand, and enters the count in the Xj register. The word entered in Xj is in positive integer format. If (Xk) is all 1's, a count of 64 decimal is entered in the Xj register. If (Xk) is all 0's, a zero word is entered in the Xj register.

This instruction is intended for use in data processing where a degree of coincidence is desired.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free five clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xk) to input register in DA module

CP2 Form partial sums

CP3 Form partial sums

CP4 Form count result in DB module
Clear Xj reservation flag

CP5 Transmit result to Xj in register modules

NOTES

1. If the j and k designators have the same value, the operand is read from and the count stored back into the same X register.

0030jjkk	Floating double precision product of (Xj) times (Xk) to Xj	[0030] [ ] [(0C)]

This instruction reads two normalized floating point operands from the Xj and Xk registers, forms a floating point double precision product, and enters the lower half of the result in the Xj register.

The two operands are unpacked from floating point format (the operands are not rounded). The exponents are added to determine the exponent for the result. The result exponent is 48 decimal less than the exponent of the upper half coefficient (the upper half coefficient is extracted with the 13XX instruction).

The coefficients are multiplied as signed integers to form a 96-bit double precision integer product. The lower half of this product is then extracted to form the 48-bit coefficient for the result. If the double precision product has only 95 significant bits, a 1-bit normalizing shift is performed before extracting the lower half, and the exponent for the result is decremented by one count.

This instruction is intended for use in multiple precision floating point calculations. Used together with the single precision FP multiply instruction 22XX, this instruction forms a 96-bit double precision product in two X registers with no loss of precision.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

EXECUTION TIMING (continued)

CP0	Instruction issues from IPT Transmit j and k designators to register modules Set Xj reservation flag
CP1	Transmit (Xj) and (Xk) to floating multiply module MA Separate exponents from coefficients
CP2	Perform sign corrections Form first 16x48 product
CP3	Form second 16x48 product
CP4	Form third 16x48 product
CP5	Merge the three 16x48 products into 96-bit result register
CP7	Clear Xj reservation flag
CP8	Enter lower 48 bits of 96 bit result and exponent result in Xj (entire 64-bit result complemented if negative)

0031 jjkk	Integer product of (Xj) times (Xk) to Xj	[0031] [ ] [(0D)]

This instruction reads two operands (limited to 48 bits plus sign and sign extension) from the Xj and Xk registers, forms the product, and delivers the result to the Xj register.

The instruction is performed in the floating multiply unit (the exponent arithmetic portion is not used).

The operands are multiplied as signed integers to form a 96-bit product. The lower 64 bits of this product are extracted and entered in the Xj register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Transmit (Xj) and (Xk) to floating multiply unit
Perform sign corrections

CP2 Form first 16x48 product

CP3 Form second 16x48 product

CP4 Form third 16x48 product

CP5 Merge the three 16x48 products into 96-bit result register

CP7 Clear Xj reservation flag

CP8 Enter lower 64 bits of 96-bit result in Xj (complemented if neg.)

0032xxxx	Program error exit	[0032] [ ] [(0E)]

This instruction format is treated as an error condition and, if executed, will set the program error exit flag in the exchange parameter word. This condition flag will then cause an exchange jump to address (XA). In this case all instructions which have issued prior to this instruction will be run to completion. Any instructions following this instruction in the current instruction word will not be executed. When all operands have arrived at the operating registers as a result of previously issued instructions, an exchange jump will occur to the exchange package which is designated by (XA).

The j and k designators in this instruction are ignored. The program address stored in the exchange package on the terminating exchange jump is advanced one count from the address of the current instruction word. This is true no matter which parcel of the current instruction word contains the program error exit instruction.

This instruction format is not intended for use in normal program code. The program error exit flag is set in the exchange parameter word (XPW) to indicate that the program may be in range but is not executing valid program code. This could occur when an incorrectly coded program jumps into an unused area of the memory field or into a data field.

| 0033xxxx | Pass

[0033]
[ ]
[(0F)]

This instruction is a 'do-nothing' instruction and is typically used to fill program instruction words where necessary to match jump destinations with word boundaries. The J and K designators are not used and are normally zero, but non-zero values will have no effect on the instruction.

ISSUE CONDITIONS

None

EXECUTION TIMING

CP0 Instruction parcel in IPT
Instruction issues

CP1 Next instruction may issue

0100jjkk	Transmit k to Xj	[0100] [ ] [(10)]

This instruction forms a 64-bit word with the 4-bit integer specified by k in the lower 4 bits of the word and 0's in the upper 60 bits. The result is entered in the Xj register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Enter k in lower 4 bits of operand register B
Clear Xj reservation flag

CP2 Enter (B) in Xj

NOTES

0101jjkk	Transmit -k to Xj	[0101] [ ] [(11)]

This instruction forms a 64-bit word with the complement of the 4-bit integer specified by k in the lower 4 bits of the word and 1's in the upper 60 bits. The result is entered in the Xj register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Enter complememt of k plus 60 bits of 1's in operand register B
Clear Xj reservation flag

CP2 Enter (B) in Xj

NOTES

0102jjkk	Integer sum of (Xj) plus k to Xj	[0102] [ ] [(12)]

This instruction forms a 64-bit sum of the operand read from the Xj register and the 4-bit integer specified by k. The result is entered in the Xj register

This instruction is intended primarily for incrementing an operand by a small number. Integer sum instruction 11XX is used for addition of larger numbers.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Enter k in lower 4 bits of operand register B

CP2 Perform partial add operation
Clear Xj reservation flag

CP3 Complete add operation and enter result in Xj

NOTES

0103jjkk	Integer difference of (Xj) minus k to Xj	[0103] [ ] [(13)]

This instruction forms the 64-bit difference of the operand read from the Xj register and the 4-bit integer specified by k. The result is entered in the Xj register.

This instruction is intended primarily for decrementing an operand by a small number. Integer difference instruction 13XX is used for subtracting larger numbers

ISSUE CONDITIONS

Xj registers is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Enter complement of k plus 60 bits of 1's in operand register B

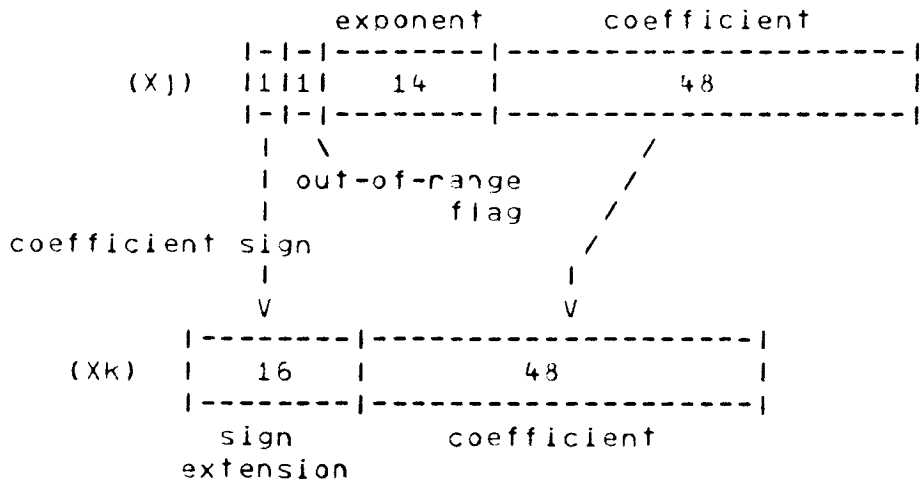
CP2 Perform partial add operation
Clear Xj reservation flag

CP3 Complete add operation and enter result in Xj

NOTES

0110jjkk	Unpack coefficient of (Xj) to Xk	[0110] [ ] [(14)]

This instruction reads a 64-bit floating point operand from the Xj register unpacks the word, and enters the 48-bit coefficient plus 16 bits extension of the coefficient sign bit in the Xk register.



No test for out-of-range condition is performed, and the out-of-range flag and exponent bits in (Xj) are replaced in (Xk) with copies of the sign bit.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xk reservation flag

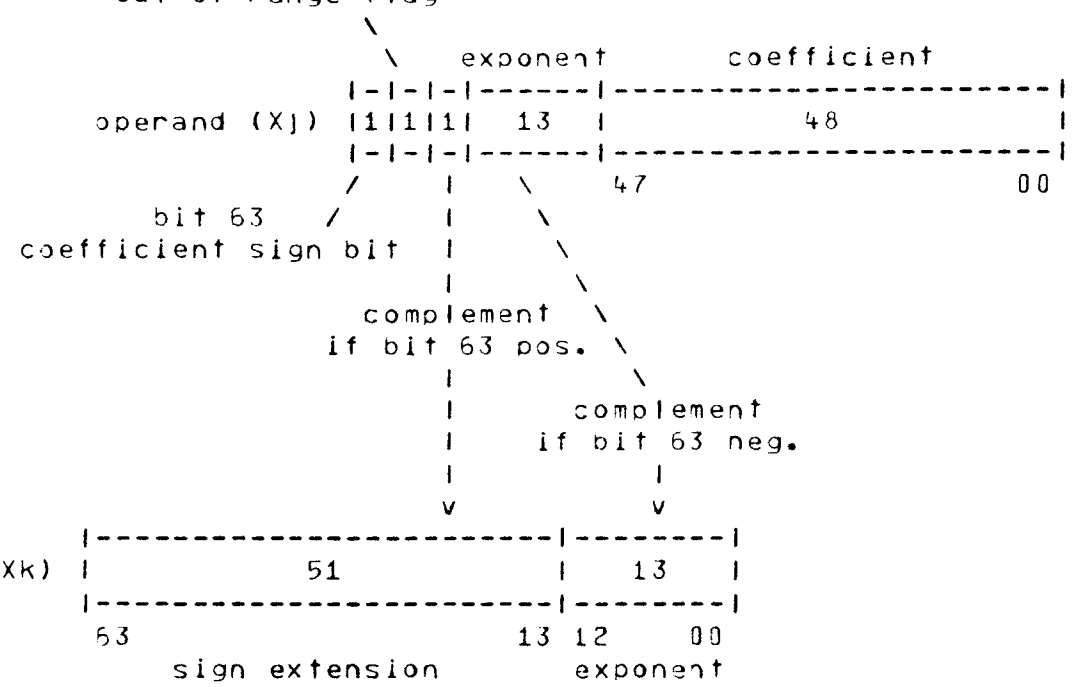
CP1 Read (Xj) to operand register A
Extract coefficient and sign
Clear Xk reservation flag

CP2 Transmit sign extended coefficient to Xk

NOTES

0111jjkk	Unpack exponent of (Xj) to Xk	[0111] [ ] [(15)]

This instruction reads a 64-bit floating point operand from the Xj register and unpacks the word. The exponent portion of (Xj) is sign extended and entered in the Xk register.



If (Xj) is positive, bits 48-60 of (Xj) are copied to Xk bits 00-12 and the complement of bit 61 of (Xj) is copied to Xk bits 13-63.

If (Xj) is negative, the complement of bits 48-63 of (Xj) are copied to Xk bits 00-12 and bit 61 of (Xj) is copied to Xk bits 13-63.

No test for out-of-range condition is performed.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT..

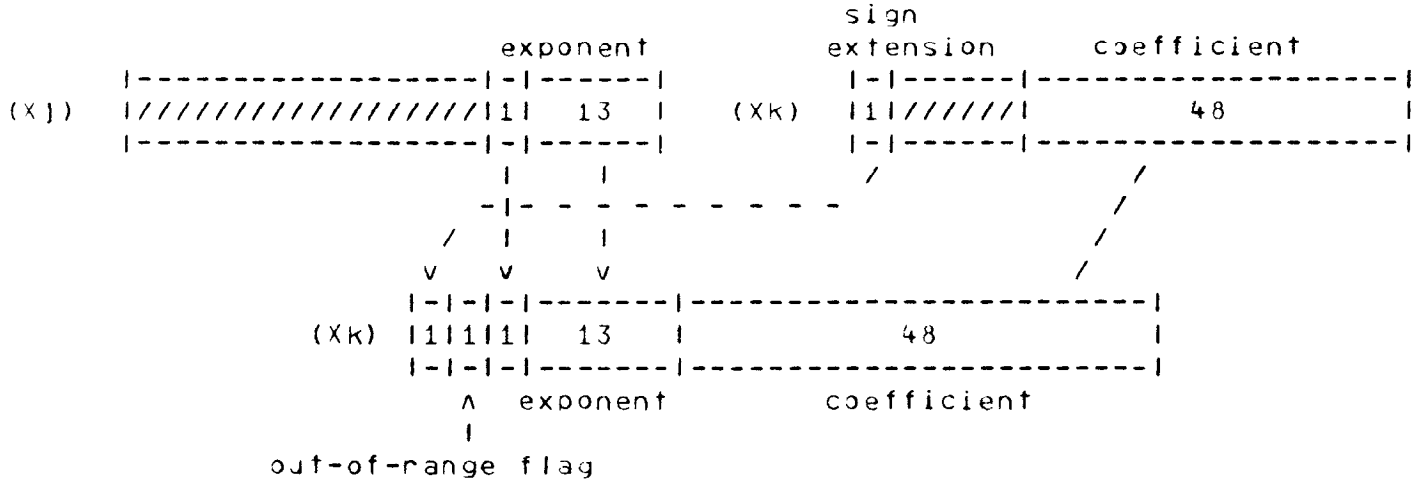
CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xk reservation flag

CP1 Read (Xj) to operand register A
Extract exponent
Clear Xk reservation flag

CP2 Transmit sign extended exponent to Xk

```
------------                                                    --------
| 0112jjkk |  Pack coefficient Xk and exponent Xj to Xk          [ 0112 ]
------------                                                    [      ]
                                                                [ (15) ]
                                                                --------
```

This instruction reads a coefficient operand from Xk and an exponent operand from Xj, packs them into a 64-bit floating point word, and enters the result in the Xk register.



To ensure that the result is normalized, this instruction must be followed by a floating point add of the result and zero (instruction 20XX).

No test for out-of-range condition is performed and the out-of-range flag bit in the result entered in (Xk) is set to the sign bit.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xk reservation flag

CP1 Read (Xj) to operand register A
Read (Xk) to operand register B
Clear Xk reservation flag

CP2 Transmit exponent of (A) and coefficient of (B) to Xk

NOTES

0113jjkk	Integer difference of zero minus (Xk) to Xj	[0113] [ ] [(17)]

This instruction forms the 64-bit difference of zero and the operand read from the (Xk) register and enters the result in the Xj register.

This instruction is intended for changing the sign of a fixed or floating point quantity. If (Xk) is zero, a positive zero will be the result.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 Enter all 0's in operand register A
Complement (Xk) and enter in operand register B

CP2 Perform partial add operation
Clear Xj reservation flag

CP3 Complete add operation and enter result in Xi

```
------------                                                    --------
| 0120xxxx |  Begin system call [MTF]                          [ 0120 ]
------------                                                   [      ]
                                                               [ (18) ]
                                                               --------
```

This instruction, when issued by a processor in monitor mode, sets the system call flag (SCF). This flag causes each processor not in monitor mode to set the system call condition in the exchange parameter word, which causes an exchange to exchange address XA.

Until an end system call instruction 0121 is executed, processors not in monitor mode will be exchanged.

This instruction will be executed only if monitor mode flag MTF in the exchange parameter word is set. If MTF is not set, the instruction is executed as a pass instruction.

This instruction is intended to be used to initiate inter-processor communication by causing processors to heed the system call for program reassignment.

ISSUE CONDITIONS

None

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT

CP1 Next instruction may issue

C

0121xxxx	End System Call [MTF]	[0121] [ ] [(19)]

This instruction, when issued by a processor in monitor mode, clears the system call flag (SCF).

This instruction will be executed only if the monitor mode flag MTF in exchange parameter word is set. If MTF is not set, the instruction is executed as a pass instruction.

The j and k designators are ignored in the instruction.

ISSUE CONDITIONS

None

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT

CP1 Next instruction may issue

```
------------                                                    --------
| 0122jjkk |  Block input channel (Xj) to address (Xk) [MTF]     [ 0122 ]
------------                                                    [      ]
                                                                [ (1A) ]
                                                                --------
```

This instruction initiates the input of a block of data arriving on I/O channel (Xj) and stores the data in consecutive address locations in memory beginning at absolute address (Xk). The length of the block of data is determined by the I/O equipment on the channel. Only the lower 4 bits of (Xj) are used to specify the channel number.

A block of data consists of one or more words sent by the I/O equipment. Each input word is either 8 or 12 bits depending on the channel selection. The input words are assembled into 64-bit words for storage in the 8600 memory. Partially assembled 64-bit words are filled out with zeros.

Sample block of sixteen assembled 8-bit words in 8600 memory:

```
              ---------------------------------------------------
(Xk)          |  0  |  1  |  2  |  3  |  4  |  5  |  6  |  7  |
              ---------------------------------------------------
(Xk)+1        |  8  |  9  | 10  |  11 |  12 | 13  | 14  | 15  |
              ---------------------------------------------------
              63                                              00
```

Sample block of sixteen assembled 12-bit words in 8600 memory:

```
              ---------------------------------------------------
(Xk)          |     0     |     1     |     2     |     3     | 4 |
              ---------------------------------------------------
(Xk)+1        |  5  |     6     |     7     |     8     |     9 | 10|
              ---------------------------------------------------
(Xk)+2        | 10  |    11     |    12     |    13     |    14 | 15|
              ---------------------------------------------------
              63                                              00
```

Each input word is accompanied by a word flag sent by the I/O equipment and is acknowledged by an 8600 channel resume signal sent back to the I/O equipment.

The block input operation initiated by this instruction is terminated by a record flag sent by the I/O equipment to the 8600. This record flag must not be sent to the 8600 by the I/O equipment until after the receipt of an 8600 channel resume signal acknowledging that the last word has been received. The record flag will cause one of the 8000 processors to be interrupted. Another 8600 channel resume will be sent to the I/O equipment to indicate that the terminating record flag has been acknowledged and that a processor has been interrupted.

This instruction will be executed only if the monitor mode flag (MTF) in the exchange parameter word is set. If the MTF is not set, the instruction is executed as a pass instruction.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues

EXECUTION TIMING

```
------------                                                    --------
| 0123JJkK |  Block output channel (XJ) from address (Xk) [MTF]  [ 0123 ]
------------                                                    [      ]
                                                                [ (1B) ]
                                                                --------
```

This instruction initiates the output of a block of data over output channel (XJ) from consecutive address locations beginning at the absolute address specified by (Xk). Only the lower 4 bits of (XJ) are used to specify the channel number.

A record flag is sent to the receiving device with the first 8-bit or 12-bit output data word.

The length of the block of data is dictated by the system I/O equipment. The channel output process is terminated by a record flag sent to the 8600 by the I/O equipment. When the record flag is received by the 8600, a parity status word is sent to the IOS indicating whether or not a memory parity error occurred while reading the output data from the 8600 memory. A parity status word of all 0's indicates that no parity error occurred; a parity status word of 1111 (octal) indicates that a parity did occur. To insure that the proper parity status word is read, the status word should not be read by the IOS until the 8600 has returned the channel resume signal acknowledging that the terminating record flag has been received and that a processor has been interrupted.

This instruction will be executed only if the monitor mode flag MTF in the exchange parameter word is set. If the monitor mode flag is not set, the instruction is executed as a pass.

ISSUE CONDITIONS

- XJ register is free one clock period after instruction issues
- Xk register is free one clock period after instruction issues

EXECUTION TIMING

```
------------                                                  --------
| 0130jjxx |  Read channel request to Xj [MTF]                [ 0130 ]
------------                                                 [      ]
                                                             [ (1C) ]
                                                             --------
```

This instruction transmits to the (Xj) register the identity number of the I/O channel which caused an interrupt. Any subsequent I/O interrupts are disabled until this instruction is executed.

The k designator is ignored in this instruction.

This instruction will be executed only if the monitor mode flag MTF in the exchange parameter word is set. If MTF is not set, the instruction will enter Xj with 0's and have no effect on channel interrupts.

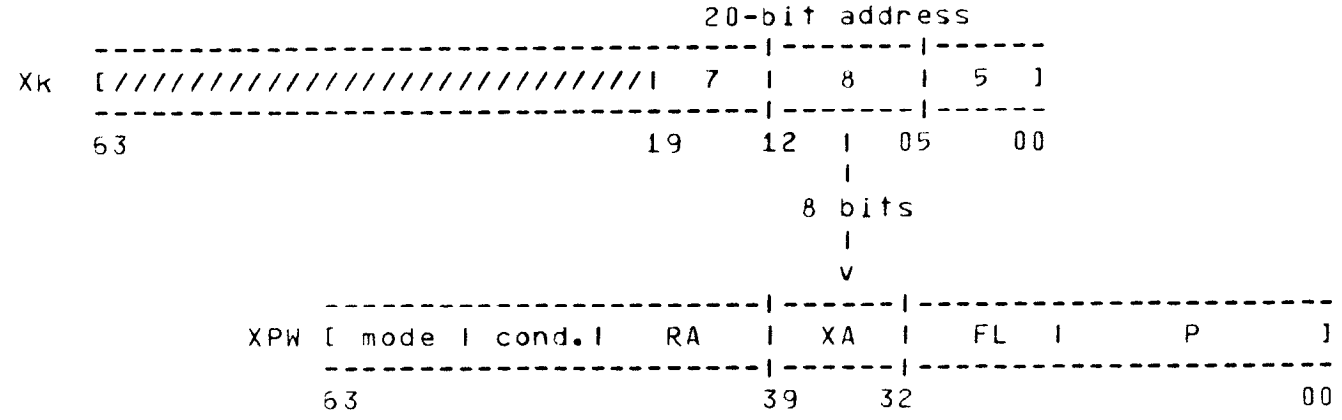
ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

0131xxkk	Load XA from (Xk) [MTF]	[0131] [ ] [(1D)]

This instruction loads the address contained in the Xk register into the exchange address XA portion of the exchange parameter word (XPW).



The j designator is ignored in this instruction.

This instruction will be executed only if the monitor mode flag MTF in the exchange parameter word is set. If MTF is not set, the instruction is executed as a pass instruction.

ISSUE CONDITIONS

Xk register is free one clock period after instruction issues

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit k designator to register modules

CP1 Read (Xk) to DA module

CP2 Transmit XA to XPW register in RC module

This instruction format is treated as an error condition and, if executed, will set the program error exit flag in the XPW register. This condition flag will then cause an exchange jump to address (XA). In this case all instructions which have issued prior to this instruction will be run to completion. Any instructions following this instruction in the current instruction word will not be executed. When all operands have arrived at the operating registers as a result of previously issued instructions, an exchange jump will occur to the exchange package designated by (XA).

The j and k designators in these instructions are ignored. The program address stored in the exchange package on the terminating exchange jump is advanced one count from the address of the current instruction word. This is true no matter which parcel of the current instruction word contains the program error exit instruction.

These instructions are not intended for use in normal program code. The program error exit flag is set in the XPW register to indicate that the program may be in range but is not executing valid program code. This could occur when an incorrectly coded program jumps into an unused area of the memory field or into a data field.

0200jjkk	kkkkkkkk	Store data at address K from Xj	[0200] [ ] [(20)]

This instruction writes one 64-bit word from the Xj register into memory at the absolute address formed by adding the address specified by K to the memory reference address RA from the processor exchange parameter word XPW.

If the field length FL is exceeded, the memory reference is aborted and the data field limit condition flag is set in the exchange parameter word XPW, and an exchange jump is made to the exchange address XA in XPW.

If a parity error occurs in reading the old data at the indicated memory address, the error is ignored and the processor operation continues in a normal manner.

This instruction allows a processor to write data into memory from any of the 16 X registers in the processor.

ISSUE CONDITIONS

Xj register is free
A storage access buffer is available for this processor

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT

CP1 Transmit 0's to operand register A
Transmit K to operand register B

CP2 Integer add (A) + (B)

0201jjkk	Store data at address (Xk) from Xj	[0201] [ ] [(21)]

This instruction writes one word from the Xj register into memory at the absolute address formed by adding the address specified in the lower 20 bits of Xk to the memory reference address RA from the processor exchange parameter word (XPW).

If the field length FL is exceeded, the memory reference is aborted, the data field limit condition flag is set in XPW, and an exchange jump is made to the exchange address XA in XPW.

If a parity error occurs in reading the old data at the indicated memory address, the error is ignored and the processor operation continues in a normal manner.

This instruction allows a processor to write data into memory from any of the 16 X registers in the processor.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT

CP1 Transmit 0's to operand register A
Transmit (Xk) to operand register B

CP2 Integer add (A) + (B)

0202jjkk	kkkkkkkk	Read data at address K to Xj and Store (Xj) into address K	[0202] [ ] [(22)]

This instruction simultaneously:

reads a word of data from an object program storage field address and enters that word in the Xj register; and

stores the original contents of the Xj register at the same object program storage field address.

The storage address is determined by adding the K field from the instruction to the object program reference address.

A separate test is made to determine if the value of the K field considered as a 20 bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word. The storage reference is aborted in this case and an exchange jump is made to exchange address XA.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

I 0203JjKk I	Read data at address (Xk) to Xj and Store (Xj) into address (Xk)	[0203] [ ] [(23)]

This instruction simultaneously:

reads a word of data from an object program storage field address and enters that word in the Xj register; and

stores the original contents of the Xj register at the same object program storage field address.

The storage address is determined by adding the current (Xj) to the object program reference address.

A separate test is made to determine if the value of (Xk) considered as a 20-bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word (XPW). The storage reference is aborted in this case and an exchange jump is made to the exchange address XA in XPW.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

```
-----------------------                                    --------
| 0210JJkk | kkkkkkkk |  Read data at address K to XJ       [ 0210 ]
-----------------------                                    [      ]
                                                           [ (24) ]
                                                           --------
```

This instruction reads a word of data from the object program storage field and enters that word in the XJ register. The storage address is determined by adding the K field from the instruction to the object program reference address.

A separate test is made to determine if the value of the K field considered as a 20-bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word (XPW). The storage reference is aborted in this case and an exchange jump is made to the exchange address XA in XPW.

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00	Instruction issues from IPT Set XJ reservation flag
CP01	Transmit zeroes to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to RA adder in IW module Transmit reference flag to SA module
CP04	Transmit lower 8 bits of address to SAC
CP05	Transmit upper bits of address to SAC
CP06	Acknowledge from SAC
CP14	Clear XJ reservation flag
CP15	Data word arrives at the X register

C

0211 jjkk	Read data at address (Xk) to Xj	[0211] [ ] [(25)]

This instruction reads a word of data from the object program storage field and enters that word in the Xj register. The storage address is determined by adding the current contents of the Xk register to the object program reference address.

A separate test is made to determine if the value of (Xk) considered as a 20-bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word. The storage reference is aborted in this case and an exchange jump is made to the exchange address XA.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00 Instruction issues fom IPT
Set Xj reservation flag

CP01 Transmit zeroes to operand register A
Transmit (Xk) to operand register B
Send destination tag to RB module

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to RA adder in IW module
Transmit reference flag to SA module

CP04 Transmit lower 8 bits of address to SAC

CP05 Transmit upper 12 bits of address to SAC

CP06 Acknowledge from SAC

CP14 Clear Xj reservation flag

CP15 Data word arrives at the X register

C

0212]]kk	kkkkkkkkk	Read program at address K to XJ	[0212] [ ] [(26)]

This instruction reads a word out of memory at absolute address K and enters that word into the XJ register. (The reference address RA and the memory field length FL from the exchange parameter word are used in this instruction only if PRF is set.)

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00	Instruction issues from IPT Set XJ reservation flag
CP01	Transmit zeroes to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to RA adder in IW module Transmit reference flag to SA module
CP04	Transmit lower 8 bits of address to SAC
CP05	Transmit upper 12 bits of address to SAC
CP06	Acknowledge from SAC
CP14	Clear XJ reservation flag
CP15	Data word arrives at the X register

C

0213jjkk	kkkkkkkkk	Read program at address (Xk) to Xj	[0213] [ ] [(27)]

This instruction reads a word out of memory at the absolute address specified by the lower 20 bits the Xk register and enters the word in the Xj register. (The memory reference address RA and the memory field length FL from the exchange parameter word are used in this instruction only if the PRF is set.)

ISSUE CONDITIONS

Xj register is free one clock after instruction issues
Xk register is free one clock after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00	Instruction issues from IPT Set Xj reservation flag
CP01	Transmit zeroes to operand register A Transmit (Xk) to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to RA adder in IW module Transmit reference flag to SA module
CP04	Transmit lower 8 bits of address to SAC Add RA or zero to address
CP05	Transmit upper 12 bits of address to SAC
CP06	Acknowledge from SAC
CP14	Clear Xj reservation flag
CP15	Data word arrives at the X register

0220jjkk | kkkkkkkk	Read program at address (P + K) to Xj	[0220] [ ] [(28)]

This instruction reads a word out of memory at an absolute address formed by adding sign extended K to the current program address P, and enters that word in the Xj register. (The memory reference address RA and the memory field length FL from the exchange parameter word are used in this instruction only if the PRF is set.)

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00	Instruction issues from IPT Set Xj reservation flag
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to RA adder in IW module Transmit reference flag to SA module
CP04	Transmit lower 8 bits of address to SAC
CP05	Transmit upper 12 bits of address to SAC
CP06	Acknowledge from SAC
CP14	Clear Xj reservation flag
CP15	Data word arrives at the X register

0221 jjkk	kkkkkkkk	Transmit P + K to Xj	[0221] [ ] [(29)]

This instruction forms the sum of the current program address P plus a sign extended increment K, and enters the result in the Xj register. The result is a 64-bit word with 0's in the upper 44 bits.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 1st instruction parcel in IPT
Instruction issues
Transmit j and k designators to register modules
Set Xj reservation flag

CP1 2nd instruction parcel in IPT
Enter P in operand register A
Transmit K to operand register B

CP2 Perform a partial add
Clear Xj reservation flag

CP3 Complete add operation and enter result in Xj

NOTES

0222jjKK | KKKKKKKK	Transmit K to XJ	[0222] [ ] [(2A)]

This instruction enters the 20-bit program constant specified by the K field in the instruction in the lower 20 bits of the XJ register. The upper bits of XJ are sign extended.

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Set XJ reservation flag

CP1 Transmit K to operand register B
Clear XJ reservation flag

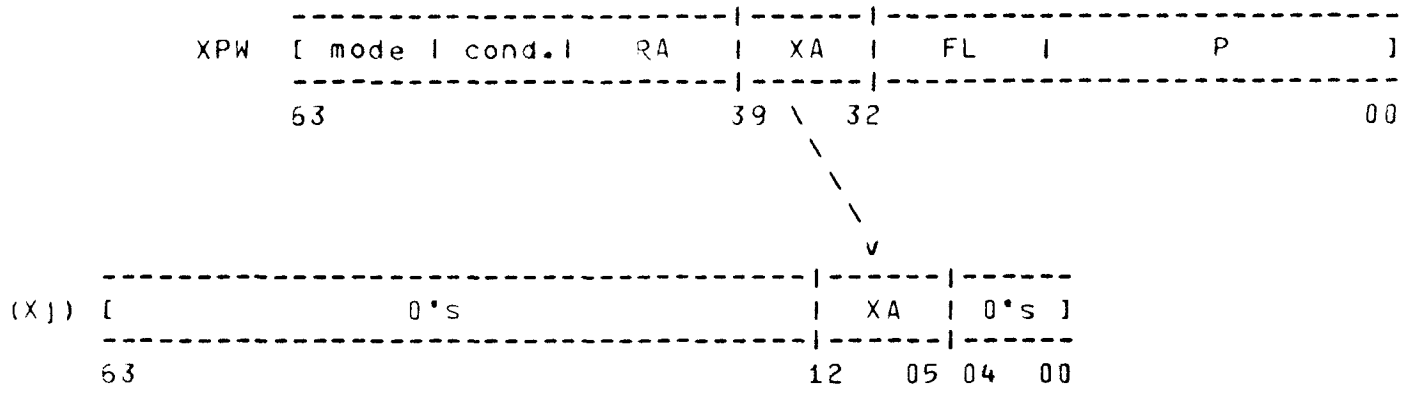
CP2 Enter (B) in XJ

NOTES

0223jj00	Transmit XA to Xj	[0223] [ ] [(28)]

This instruction enters the 8-bit exchange address (XA) portion of the exchange parameter word (XPW) in the Xj register. The k designator should be zero in this instruction.

The XA portion of XPW contains the exchange package address with the lowest order 5 bits removed. These lowest order 5 bits of 0's are added in this instruction when XA is transmitted to Xj.



ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Set Xj reservation flag

CP1 Transmit XA to operand register B
Clear Xj reservation flag

CP2 Enter (B) in Xj

NOTES

1. If the k designator in this instruction is other than zero, the upper 44 bits of Xj may be 1's since a copy of the upper bit of the k designator is placed in the upper 44 bits of Xj.

0230xxkk	Set interlock flags from (Xk) [IPF]	[0230] [ ] [(20)]

This instruction sets the interlock flags specified by the lower 20 bits of (Xk). '1' bits in lower 20 bits of (Xk) set the corresponding bits in the 20-bit interlock flag register. '0' bits in (Xk) do not change the contents of the interlock flag register.

The j designator and the upper 44 bits of Xk are ignored in this instruction

This instruction will be executed only if the interlock flag IPF in the exchange parameter word is set. If IPF is not set, the instruction is executed as a pass instruction.

```
------------                                                    --------
| 0231xxkk |  Clear interlock from (Xk)    [IPF]                [ 0231 ]
------------                                                   [      ]
                                                               [ (20) ]
                                                               --------
```

This instruction clears any of the 20 interlock flags specified by the lower 20 bits of (Xk). '1' bits in the lower 20 bits of (Xk) clear the corresponding bits in the 20-bit interlock flag register. '0' bits in (Xk) do not change the contents of the interlock flag register.

The j designator and the upper 44 bits of (Xk) are ignored in this instruction.

This instruction will be executed only if the interlock flag IPF in the exchange parameter word is set. If IPF is not set, the instruction will be executed as a pass instruction.

0232jj00 i	Read interlock flags to (Xj)

0232
(2E)

This instruction enters the contents of the 20-bit interlock register in the lower 20 bits of the Xj register. 0's are entered in the upper 44 bits of Xj.

The k designator should be zero in this instruction.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT
Set Xj reservation flag

CP1 Transmit interlock register to operand register B
Clear Xj reservation flag

CP2 Enter (B) in Xj

NOTES

1. If the k designator in this instruction is not zero, the upper 44 bits of Xj may be 1's since the upper bit of the k designator is copied into the upper 44 bits of Xj.

0233jj00	Read internal clock to Xj	[0233] [ ] [(2F)]

This instruction enters the 20-bit current contents of the internal real time clock counter into the lower 20 bits of the Xj register. 0's are entered in the upper 44 bits of Xj.

The k designator should be zero in this instruction.

This instruction is intended primarily for use in determining the elapsed time between selected points in program execution.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Set Xj reservation flag

CP1 Transmit RTC to operand register B
Clear Xj reservation flag

CP2 Transmit operand register B to Xj register

NOTES

1. If the k designator in this instruction is not zero, the upper 44 bits of Xj may be 1's since the upper bit of the k designator is copied into the upper 44 bits of Xj.

C

```
----------------------
| 0300jjkk | kkkkkkkk |  Jump to P + K
----------------------
```

```
--------
[ 0300 ]
[      ]
[ (30) ]
--------
```

This instruction terminates the current program sequence and initiates a new sequence. The value of the K field from the instruction considered as a 20-bit positive integer is added to the current contents of the P register. No further instructions are issued from the current instruction word. If the instruction address stack IAS contains an address equal to the new content of the P register, the corresponding instruction word is read to the current instruction word register CIW. If there is no address coincidence in the IAS an instruction fetch is initiated for the new program sequence.

ISSUE CONDITIONS

none

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	Coincidence in IAS
CP05	Read IWS to CIW register
CP06	Read CIW register to IPT
CP07	Next instruction can issue

BRANCH OUT OF STACK

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	No coincidence in IAS Set out of stack flag OSF
CP05	Transmit P to RA adder Transmit reference flag to storage access control
CP06	Lower 8 bits of address to storage access control Transmit P to IFA adder Transmit P to NSA register
CP07	Upper 12 bits of address to storage access control
CP08	Acknowledge from storage access control
CP16	Transmit NSA to IAS rank 11
CP17	Instruction word arrives at IWS
CP18	Read IWS to CIW register
CP19	Read CIW register to IPT
CP20	Next instruction can issue

C

```
-----------------------
| 0301jjkk | kkkkkkkk |  Call subroutine at P + K
-----------------------
```

```
--------
[ 0301 ]
[      ]
[ (31) ]
--------
```

This instruction enters the current program address P in the Xj register and causes the current program address sequence to unconditionally branch to a new program address sequence beginning at the address formed by adding the value of the K field from the instruction considered as a 20-bit positive integer to the current contents of the P register. No further instructions are issued from the current instruction word. If the instruction address stack IAS contains an address equal to the new content of the P register, the corresponding instruction word is read to the current instruction word register CIW. If there is no address coincidence in the IAS an instruction fetch is initiated for the new program sequence.

The return address is entered in the Xj register by this instruction for use by the subroutine on completion.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free 3 clock periods after issue

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to XJ

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction can issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to XJ

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to NSA register
Transmit P to IFA register
Abort IWS rank 04-11 shift

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS
P/IAS rank 11 coincidence

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction can issue

0302jjkk	kkkkkkkk	Jump to P + K if (Xj) in range	[0302] [ ] [(32)]

This instruction causes the program sequence to branch to the current program address P plus a sign extended increment K if (Xj) is in range, or continue the current program address sequence if (Xj) is not in range. (Xj) is not in range for any of the following cases:

(Xj) =	0111 1111 1111 xxxx xxxx	positive overflow
(Xj) =	1000 0000 0000 xxxx xxxx	negative overflow
(Xj) =	0011 1111 1111 xxxx xxxx	positive indefinite
(Xj) =	1100 0000 0000 xxxx xxxx	negative indefinite
(Xj) =	01xx xxxx xxxx xxxx xxxx	positive out of range
(Xj) =	10xx xxxx xxxx xxxx xxxx	negative out of range

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) is not in range (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00 Instruction issues from IPT
Transmit j designator to register modules

CP01 Transmit P to operand register A
Transmit K to operand register B
Transmit (Xj) flags to IA module

CP02 Begin integer add of (A) and (B)
Branch condition not satisfied

CP03 Next instruction may issue

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to IFA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

```
----------------------                                          --------
| 0303jjkk | kkkkkkkk |  Jump to P + K if (Xj) not in range     [ 0303 ]
----------------------                                          [      ]
                                                                [ (33) ]
                                                                --------
```

This instruction causes the program sequence to branch to the current program address P plus a sign extended increment K if (Xj) is not in range, or continue the current program address sequence if (Xj) is in range. (Xj) is not in range for any of the following cases:

(Xj) =	0111 1111 1111 xxxx xxxx	positive integer
(Xj) =	1000 0000 0000 xxxx xxxx	negative overflow
(Xj) =	0011 1111 1111 xxxx xxxx	positive indefinite
(Xj) =	1100 0000 0000 xxxx xxxx	negative indefinite
(Xj) =	01xx xxxx xxxx xxxx xxxx	positive out of range
(Xj) =	10xx xxxx xxxx xxxx xxxx	negative out of range

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) is in range (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00 Instruction issues from IPT
Transmit j designator to register modules

CP01 Transmit P to operand register A
Transmit K to operand register B
Transmit (Xj) flags to IA module

CP02 Begin integer add of (A) + (B)
Branch condition not satisfied

CP03 Next instruction may issue

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

0310 JJ kk	kkkkkkkkk	Jump to P + K if (XJ) equal to 0	[0310] [ ] [(34)]

This instruction causes the program sequence to terminate and branch to the current program address P plus a sign extended increment K if (XJ) is equal to 0, or continue the current program address sequence if (XJ) is not equal to 0. (XJ) is equal to 0 for both of the following cases:

```
(XJ)  =  00000000 ..... 00000000   plus zero
(XJ)  =  11111111 ..... 11111111   minus zero
```

ISSUE CONDITIONS

XJ register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (XJ) is not equal to zero (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00 Instruction issues from IPT
Transmit J designator to register modules

CP01 Transmit P to operand register A
Transmit K to operand register B
Perform partial zero test in register modules

CP02 Begin integer add of (A) + (B)
Transmit partial zero test to IA module
Complete zero test
Branch condition not satisfied

CP03 Next instruction may issue

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

0311jjkk	kkkkkkkkk	Jump to P + K if (Xj) not equal to 0	[0311] [ ] [(35)]

This instruction causes the program sequence to terminate and branch to the current program address P plus a sign extended increment K if (Xj) is not equal to 0, or continue the current program address sequence if (Xj) is equal to zero. (Xj) is equal to 0 for both of the following cases:

```
(Xj)  =  00000000 ..... 00000000   plus zero
(Xj)  =  11111111 ..... 11111111   minus zero
```

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) is equal to zero (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Begin integer add Branch condition not satisfied
CP03	Next instruction may issue

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	Coincidence in IAS
CP05	Read IWS to CIW register
CP06	Read CIW register to IPT
CP07	Next instruction may issue

BRANCH OUT OF STACK

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	No coincidence in IAS Set out of stack flag OSF
CP05	Transmit P to RA adder Transmit reference flag to storage access control
CP06	Lower 8 bits of address to storage access control Transmit P to IFA register Transmit P to NSA register
CP07	Upper 12 bits of address to storage access control
CP08	Acknowledge from storage access control
CP17	Instruction word arrives at IWS
CP18	Read IWS to CIW register
CP19	Read CIW register to IPT
CP20	Next instruction may issue

C

0312jjkk | kkkkkkkk | Jump to P + K if (Xj) is positive	[0312] [ ] [(36)]

This instruction causes the program sequence to terminate and branch to the current program address P plus a sign extended increment K if (Xj) is positive, or continue the current program address sequence if (Xj) is negative.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) is negative (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Begin integer add Branch condition not satisfied
CP03	Next instruction may issue

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

C

0313jjkk	kkkkkkkk	Jump to P + K if (Xj) is negative	[0313] [ ] [(37)]

This instruction causes the program sequence to terminate and branch to the current program address P plus a sign extended increment K if (Xj) is negative, or continue the current program address sequence if (Xj) is positive.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) is positive (branch fall through), execution time is three clock periods

BRANCH FALL THROUGH

CP00	Instruction issues from IPT
CP01	Transmit P to operand register A Transmit K to operand register B
CP02	Begin integer add Branch condition not satisfied
CP03	Next instruction may issue

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit P to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

```
-----------------------                                        --------
| 0320 | KK | KKKKKKKK |  Call subroutine at address K          [ 0320 ]
-----------------------                                        [      ]
                                                               [ (38) ]
                                                               --------
```

This instruction enters the current program address P in the XJ register and causes the current program address sequence to unconditionally branch to a new program address sequence beginning at the address specified by the K field from the instruction.

ISSUE CONDITIONS

XJ register is free one clock period after issue

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit zeros to operand register A
Transmit K to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to XJ

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit zeros to operand register A
Transmit K to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to XJ

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS
P/IAS rank 11 coincidence

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

C

0321jjkk	Call subroutine at address (Xk)	[0321] [ ] [(39)]

This instruction enters the current program address P in the Xj register and causes the current program address sequence to unconditionally branch to a new program address sequence beginning at address specified by the contents of the Xk register.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

A

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit zeros to operand register A
Transmit (Xk) to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to Xj

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit zeros to operand register A
Transmit (Xk) to operand register B

CP02 Integer add front half
Transmit P & PRF to operand register A

CP03 Integer add back half
Transmit integer sum to P register
Transmit (A) to Xj

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS
P/IAS rank 11 coincidence

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

0322	jjkk | kkkkkkkkk	Call library routine at address K [Clear PRF]	0322 (3A)

This instruction clears the program reference flag PRF. The XJ register is cleared and entered with the current contents of the P register. The P register is cleared and entered with the value of the K field. The IAS is cleared so that all address registers contain 3333333333. No further instructions are issued from the current instruction word. Any instruction fetches in process are discarded on arrival. A new instruction fetch is initiated for the absolute address now contained in the P register.

This instruction is intended for use by an object program in calling a resident library routine which is outside of the object program field. The K field in the instruction contains the absolute address of the library routine entrance. The return address is entered in the XJ register by this instruction for use by the library routine on completion.

The program reference flag PRF is cleared by this instruction so that the resident library routine code can be executed directly from the resident locations. All instructions except the ten data storage reference group instructions are executed in absolute mode.

The instruction address stack IAS is cleared to avoid conflicts between the old object program which is relative to RA and the library routine addresses which are absolute.

ISSUE CONDITIONS

XJ register is free one clock period after instruction issue
X register input path is free three clock periods after issue

EXECUTION TIMING

Minimum execution time for this instruction is 18 clock periods. Delays may occur in the arrival of the new instruction word at the processor IWS due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

CP00	Instruction issues from IPT
CP01	Transmit zeros to operand register A Transmit K to operand register B Clear program reference flag PRF
CP02	Integer add front half Transmit P to operand register A Clear instruction address stack IAS Abort fetches in process
CP03	Integer add back half Transmit operand register A to Xj register Transmit integer sum to P register
CP04	No P/IAS coincidence Set out-of-stack flag OSF
CP05	Transmit reference flag to SA module
CP06	Lower 8 bits of address to SA module Transmit P to NSA Transmit P to IFA
CP07	Upper 12 bits of address to SA module
CP08	Acknowledge from SA module
CP16	Enter address in IAS
CP17	Instruction word arrives at IWS
CP18	Read IWS to CIW register
CP19	Read CIW to IPT
CP20	Next instruction may issue

0323jjkk	Call library routine at address (Xk) [Clear PRF]	[0323] [ ] [(38)]

This instruction clears the program reference flag PRF. The X register indicated by the j designator is cleared and entered with the current contents of the P register. The P register is cleared and entered with the current contents of the X register indicated by the k designator. The instruction address stack IAS is cleared so that all address registers contain 3333333333. No further instructions are issued from the current instruction word. Any instruction fetches in process are discarded on arrival. A new instruction fetch is initiated for the absolute address now contained in the P register.

This instruction is intended for use by an object program in calling a resident library routine which is outside of the object program field. The Xk register contains the absolute address of the library routine entrance. The return address is entered in the Xj register by this instruction for use by the library routine on completion.

The program reference flag PRF is cleared by this instruction so that the resident library routine code can be executed directly from the resident locations. All instructions with the exception of the ten data storage reference instructions are executed in absolute mode.

The instruction address stack IAS is cleared to avoid possible conflicts between the old object program code which is relative to RA and the library routine addresses which are absolute.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issue
Xk register is free one clock period after instruction issue
X register input path is free three clock periods after issue

EXECUTION TIMING

Minimum execution time for this instruction is 18 clock periods. Delays may occur in the arrival of the new instruction word at the processor IWS due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

CP00	Instruction issues from IPT
CP01	Transmit zeros to operand register A Transmit Xk to operand register B Clear program reference flag PRF
CP02	Integer add front half Transmit P & PRF to operand register A Clear instruction address stack IAS Abort fetches in process
CP03	Integer add back half Transmit operand register A to Xj register Transmit integer sum to P register
CP04	No coincidence in IAS Set out-of-stack flag OSF
CP05	Transmit reference flag to SA module
CP06	Lower 8 bits of address to SA module Transmit P to IFA register Transmit P to NSA register
CP07	Upper 12 bits of address to SA module
CP08	Acknowledge from SA module
CP16	Transmit NSA to IAS
CP17	Instruction word arrives at IWS
CP18	Read IWS to CIW register
CP19	Read CIW to IPT
CP20	Next instruction may issue

0330jjkk	Subroutine exit, computed jump to (Xj) + k	[0330] [ ] [(3C)]

This instruction terminates the current program sequence and initiates a new sequence. The P register is cleared and entered with the integer sum of the content of the X register indicated by the j designator and the value of the k designator considered as a 4-bit positive integer. No further instructions are issued from the current instruction word. If the instruction address stack IAS contains an address equal to the new content of the P register, the corresponding instruction word is read to the current instruction word register CIW. If there is no address coincidence in the IAS an instruction fetch is initiated for the new program sequence.

This instruction is intended for use by a subroutine in returning to a calling program. It may also be used as a normal unconditional jump to a computed destination. The k designator is added to the content of the X register to provide a convenient vehicle for passing over error exit instructions when a subroutine is returning to a calling program.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit XJ to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit XJ to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

C

0331jjkk	Library routine exit to (Xj) + k [set/clear PRF]	[0331] [ ] [(3D)]

This instruction sets or clears the program reference flag PRF. Bit 52 of (Xj) causes the PRF to be set if it is a "1" or cleared if it is a "0".

The P register is cleared and entered with the integer sum of the contents of the Xj register and the value of the k designator considered as a 4-bit positive integer. The instruction address stack (IAS) is cleared so that all address registers contain dibits 3333333333. No further instructions are issued from the current instruction word. Any instruction fetches in process are discarded on arrival. A new instruction fetch is initiated for the relative address now contained in the P register.

This instruction is intended for use by a library routine outside of the object program field. The Xj register contains the return address of the calling object program relative to the object program reference address RA. The k designator is added to the object program return address to provide a convenient vehicle for passing over error exit branch instructions which may be encoded in the object program following the library routine calling instruction. Various error exits in the library routine may use different k values to select the various error exit branch instructions

The program reference flag PRF is set by this instruction so that all future instruction fetches will be relative to RA. The instruction address stack IAS is cleared to avoid possible conflicts between the old library routine addresses which are absolute, and the new object program addresses which are relative.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issue

EXECUTION TIMING

Minimum execution time for this instruction is 18 clock periods. Delays may occur in the arrival of the new instruction word at the processor IWS due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

CP00	Instruction issues from IPT
CP01	Transmit Xj to operand register A Transmit k to operand register B Set PRF to bit 62 of Xj Clear IAS
CP02	Integer add front half Clear instruction address stack IAS Abort fetches in process
CP03	Integer add back half Transmit integer sum to P register Transmit integer sum to RA adder
CP04	No coincidence in IAS Set out-of-stack flag OSF
CP05	Transmit reference flag to storage access control
CP06	Lower 8 bits of address to storage access control Add RA to integer sum
CP07	Upper 12 bits of address to storage access control
CP08	Acknowledge from storage access control
CP16	Transmit NSA to IAS
CP17	Instruction word arrives at IWS Advance NSA
CP18	Read IWS to CIW register
CP19	Read CIW register to IPT
CP20	Next instruction may issue

0332JJkk | kkkkkkkk	Jump to K	[0322] [ ] [(3E)]

This instruction causes the current program address sequence to unconditionally branch to a new program address sequence beginning at the address specified by the value of the K field from the instruction. No further instructions are issued from the current instruction word. If the instruction address stack IAS contains an address equal to the new content of the P register, the corresponding instruction word is read to the current instruction word register CIW. If there is no address coincidence in the IAS an instruction fetch is initiated for the new program sequence.

ISSUE CONDITIONS

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control.

EXECUTION TIMING (continued)

BRANCH IN STACK

CP00	Instruction issues from IPT
CPp1	Transmit zeros to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	Coincidence in IAS
CP05	Read IWS to CIW register
CP06	Read CIW register to IPT
CP07	Next instruction may issue

BRANCH OUT OF STACK

CP00	Instruction issues from IPT
CP01	Transmit zeros to operand register A Transmit K to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to P register
CP04	No coincidence in IAS Set out of stack flag OSF
CP05	Transmit P to RA adder Transmit reference flag to storage access control
CP06	Lower 8 bits of address to storage access control Transmit P to IFA register Transmit P to NSA register
CP07	Upper 12 bits of address to storage access control
CP08	Acknowledge from storage access control
CP16	Transmit NSA to IAS rank 11
CP17	Instruction word arrives at IWS
CP18	Read IWS to CIW register
CP19	Read CIW register to IPT
CP20	Next instruction may issue

C

```
------------                                                    --------
| 0333xxxx |  Exchange Exit                                     [ 0333 ]
------------                                                    [      ]
                                                                [ (3F) ]
                                                                --------
```

This instruction causes the current program sequence to terminate with an exchange jump to address (XA) which is the absolute address of an exchange package. The lowest order 5 bits have been removed from (XA) because exchange packages reside in low storage addresses at address multiples of 32.

The j and k designators in this instruction are ignored. The program address stored in the exchange package in the terminating exchange jump is advanced one count from the address of the current instruction word. This is true no matter which parcel of the current instruction word contains the exchange exit instruction.

ISSUE CONDITIONS

None

EXECUTION TIMING

CP00 Instruction issues from IPT

CP01 Exchange exit flag set in XPW in RD module

CP02 Set interrupt flag in RD module

CP03 Interrupt condition sent to IA module

CP04 Clear CIW and IPT
Clear IAS
Abort fetches in progress

CP05 Exchange sequence flag set - may be delayed waiting for:
1) all clear registers; 2) no fetch in progress
All 1's to NSA
Transmit XA to IFA

CP06 Set storage sequence CP1 for XPW

CP07 Set storage sequence CP2 for XPW
Transmit exchange count as read/write tag to RB module
Transmit exchange count as address to IW module
Advance exchange count
Set one storage reservation

CP08 Set storage sequence CP3 for XPW

CP09 Send reference flag to SA module for XPW
Set storage sequence CP1 for X register 33

CP10 Send lower 8 bits of address to SA module
Set storage sequence CP2 for X register 33
Transmit exchange count as read/write tag to RB module
Transmit exchange count as address to IW module
Advance exchange count
Set two storage reservations (block go exchange)

CP11 Send upper 12 bits of address to SA module
Set storage sequence CP3 for X register 33

CP12 Acknowledge from SA module (may be delayed by conflicts)
Send reference flag to SA module for X register 33

CP13 Send lower 8 bits of address to SA module
Set storage sequence CP1 for X register 32
Clear two storage reservations flag

The sequence continues as above for the other registers. The setting of storage sequence CP1 alternates between 3 clock periods and 4 clock periods for adjacent registers.

XPW or X Register	Set storage sequence CP1	Enter word into register
XPW	CP06	CP21
33 (F)	09	24
32 (E)	13	28
31 (D)	16	31
30 (C)	20	35
23 (B)	23	38
22 (A)	27	42
21 (9)	30	45
20 (8)	34	49
13 (7)	37	52
12 (6)	41	56
11 (5)	44	59
10 (4)	48	63
03 (3)	51	66
02 (2)	55	70
01 (1)	58	73
00 (0)	62	77

CP62 Set storage sequence CP1 for X register 00

CP63 Enter storage word into X register 10
Set storage sequence CP2 for X register 00
Transmit exchange count as read/write tag to RB module
Transmit exchange count as address to IW module
Advance exchange count

CP64 Set storage sequence CP3 for X register 00

CP65 Send reference flag to SA module for X register 00
End exchange sequence

CP66 Send lower 8 bits of address to SA module
Enter memory word into X register 03
No P/IAS coincidence
Set out-of-stack flag

CP67 Send upper 12 bits of address to SA module for X register 00
Send fetch reference flag to SA module

CP68 Acknowledge for X register 00 from SA module
Send lower 8 bits of fetch address to SA module

CP69 Send upper 12 bits of fetch address to SA module

CP70 Enter memory word into X register 02
Acknowledge for fetch from SA module

CP73 Enter memory word into X register 01

CP77 Enter memory word into X register 00

CP78 NSA to IAS rank 11

CP79 Storage to IWS
P/IAS rank 11 coincidence

CP80 IWS to CIW

CP81 CIW to IPT

CP82 Next instruction may issue

100njjnn	Save lower (Xj) for n bits	[100X] [ ] [(40)]

This instruction reads a 64-bit operand from the Xj register, forms a mask (1's in the lower n bits, 0's in the upper bits), and performs a bit-by-bit logical product of (Xj) and the mask. The result is entered in the Xj register. A sample of the instruction operation is listed below in binary notation.

Sample: n = 000100 [n]

mask	=	0000 0000	1111
64-bit (Xj) operand	=	1011 1010	1101
64-bit (Xj) result	=	0000 0000	1101

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free two clock periods after issue

EXECUTION TIMING

CP0 Instruction issues from IPT
Transmit j and n designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Enter 1's in lower n bits of operand register B
Clear Xj reservation flag

CP2 Enter logical product of (A) and (B) in Xj

NOTES

1. The maximum number of mask bits which may be specified by n is 63 decimal.

101njjnn	Blank lower (Xj) for n bits	[101X] [ ] [(44)]

This instruction reads a 64-bit operand from the Xj register, forms a mask (1's in the lower n bits, 0's in the upper bits), complements the mask, and performs a bit-by-bit logical product of (Xj) and the complemented mask. The result is entered in the Xj register. A sample of the instruction operation is listed below in binary notation.

Sample: (64 bits)

n = 111100 [<-------- n -------->]

mask	=	0000	1111 1111 1111
complemented mask	=	1111	0000 0000 0000
(Xj) initial	=	1011	1100 1010 1101
(Xj) terminal	=	1011	0000 0000 0000

ISSUE CONDITIONS

Xj register is free one
X register input path is free two clock periods after issue

EXECUTION TIMING

CP0 Instruction issues from IPT
Transmit j and n designators to register modules
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Enter 1's in lower n bits of operand register B
Clear Xj reservation flag

CP2 Enter logical product of (A) and (B) complement in Xj

NOTES

1. The maximum number of mask bits which may be specified by n is 63 decimal.

102njjnn	Left shift (Xj) by n bits (circular)	[102X] [ ] [(48)]

This instruction reads a 64-bit operand from the Xj register, shifts the operand left circularly by n bit positions, and writes the word back into the Xj register. n is a 6-bit positive integer operand with the 2-bit i designator as the upper 2 bits and the k designator as the lower 4 bits. In the example below, the n designator is 000100.

```
Sample (64 bits):   (Xj) operand  =  10110000......00000000
                    (Xj) result   =  00000000......00001011
```

In a left circular shift operation, each bit shifted off the upper end of the 64-bit word is inserted in the lowest order bit position.

This instruction is intended for use in data processing as distinguished from numerical computation, and is used whenever a data word is to be shifted to the left by a predetermined number of places. If the shift count is derived in the execution of a program, instructions 0012 or 0013 should be used.

ISSUE CONDITIONS

Xj register is free one clock period after issue
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Clear Xj reservation flag

CP2 Begin operand shift

CP3 Complete operand shift
Transmit operand to Xj

NOTES

1. The maximum shift count which may be specified by (Xj) is 63 decimal.

2. If the shift count is 0 this instruction reads the operand from register Xj and returns it unaltered to register Xj. The timing for this case is the same as for the general case.

3. If the operand bits are all 1's or all 0's they are treated in the same manner as any other bit pattern and the timing is the same as for the general case.

103njjnn	Right shift (Xj) by n bits (with sign extension)	[103X] [ ] [(4C)]

This instruction causes the shift unit to read a 64-bit operand from the Xj register to operand register A, shift the operand right with sign extension by n bit positions, and write the word back into the Xj register. n is a 6-bit positive integer operand with the 2-bit i designator as the upper 2 bits and the k designator as the lower 4 bits. In the sample operations below the n designators are 111010 and 000011 binary.

```
Sample (64 bits):      (Xj) operand  =  01000000......00000111
   n = 111010          (Xj) result   =  00000000......00010000

Sample (64 bits):      (Xj) operand  =  11000000......00100010
   n = 000011          (Xj) result   =  11111000......00000100
```

Each bit shifted off the lower end of the 64-bit word is discarded and the highest order bit is replaced with a copy of the original operand sign bit.

This instruction is used whenever a data word is to be shifted right with sign extension by a predetermined number of places. If the shift count is derived in the execution of a program, instruction 0012 or 0013 should be used.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Set Xj reservation flag

CP1 Read (Xj) to operand register A
Clear Xj reservation flag

CP2 Begin operand shift

CP3 Complete operand shift
Transmit operand to Xj

NOTES

1. The maximum shift count which may be specified by (Xj) is 63 decimal.

1 11iiJJkk 1 kkkkkkkkk 1	Integer sum of (XJ) plus K to Xi	[11XX] [ ] [(5x)]

This instruction forms the 1's complement sum of the 64-bit operand read from the XJ register and the integer specified by sign extended K. The result is entered in the Xi register. An overflow condition is ignored.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
XJ register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit J and k designators to register modules
Set Xi reservation flag

CP1 Enter (XJ) in operand register A
Enter K in operand register B

CP2 Perform add operation
Clear Xi reservation flag

CP3 Enter result in Xi

NOTES

12iijjkk	Integer sum of (Xj) plus (Xk) to Xi	[12XX] [ ] [(6x)]

This instruction forms a 64-bit 1's complement sum of the operands read from the Xj and Xk registers and enters the result in the Xi register. The operands are assumed to be signed integers. An overflow condition is ignored.

This instruction is intended for the addition of integers and is also useful in merging and comparing data fields during data processing.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit i, j and k designators to register modules
Set Xi reservation flag

CP1 Enter (Xj) in operand register A
Enter (Xk) in operand register B

CP2 Perform add operation
Clear Xi reservation flag

CP3 Enter result in Xi

NOTES

1. If the j and k designators have the same value, the designated 64-bit operand is added to itself and the resulting sum entered in the Xi register.

2. If the i designator has the same value as the j designator or the k designator, this instruction becomes a replace add instruction. The initial (Xi) is added to the other operand and the result then stored back in the Xi register.

I 13iijjkk I	Integer difference of (Xj) minus (Xk) to Xi	[13XX] [ ] [(7x)]

This instruction forms the 64-bit 1's complement difference of the operands read from the Xj and Xk registers and enters the result of (Xj) minus (Xk) in the Xi register. The operands are assumed to be signed integers. An overflow condition is ignored.

This instruction is intended for subtraction of integers and is also useful in comparing data fields during data processing.

ISSUE CONDITIONS

Xi register is free one clock after instruction issues
Xj register is free one clock after instruction issues
Xk register is free one clock after instruction issues
X register input path is free three clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT

CP0 Instruction issues from IPT
Transmit i, j and k designators to register modules
Set Xi reservation flag

CP1 Enter (Xj) in operand register A
Complement (Xk) and enter in operand register B

CP2 Perform add operation
Clear Xi reservation flag

CP3 Enter result in Xi

NOTES

1. If the j and k designators have the same value, the designated 64-bit operand is subtracted from itself. The result is a positive zero entered in the Xi register.

2. If the i designator has the same value as the j designator or the k designator, this instruction becomes a replace subtract instruction. The initial (Xi) is read as an operand, and the resulting difference is then stored in the same register.

20iijjkk	Floating sum of (Xj) plus (Xk) to Xi	[20XX] [ ] [(8x)]

This instruction forms the double precision sum of two floating point operands read from the Xj and Xk registers and enters the normalized single precision upper half of the result in the Xi register.

The operands are not rounded in this operation and may or may not be normalized. They are unpacked from floating point format and the exponents compared. The coefficient with the smaller exponent is shifted down by the difference of the exponents so as to align bits of corresponding significance, and a 97-bit adder forms a double precision 1's complement sum. A 48-bit result coefficient is read from the upper half of this sum.

If an overflow of the highest order coefficient bit occurs during the addition process, the result coefficient is displaced one bit and the result exponent is corrected by one count. The upper half result entered in the Xi register is normalized.

This instruction is intended for use in floating point calculations where rounding of operands is not desired. This is the case in multiple precision arithmetic and in calculation involving error analysis.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

EXECUTION TIMING (continued)

CP0	Instruction issues from IPT Transmit i, j and k designators to register modules Set Xj reservation flag
CP1	Read (Xj) to floating add module FA Read (Xk) to floating add module FA Compare exponents Transmit coefficients to pre-add shift register
CP2	Select smaller exponent
CP3	Shift coefficients for bit alignment Transmit coefficients to 97-bit adder
CP4	Form double precision sum
CP5	Transmit DP sum to bit position network
CP6	Determine significant bit position Transmit result to shift network
CP7	Perform 96-bit normalize shift Clear Xj reservation flag
CP8	Enter upper half of result in Xi

21iijjkk	Floating difference of (Xj) minus (Xk) to Xi	[21XX] [ ] [(9x)]

This instruction forms the floating point difference of two floating point operands read from the Xj and Xk registers and enters the normalized result of (Xj) minus (Xk) in the Xi register. The result entered in Xi is the upper half of a double precision number.

The operands are not rounded in this operation and may or may not be normalized. (Xk) is complemented. (Xj) and the complemented (Xk) are unpacked from floating point format. The exponents are compared and the coefficient with the smaller exponent is shifted down by the difference of the exponents so as to align bits of corresponding significance. A 97-bit adder then forms a double precision 1's complement sum and a 48-bit result coefficient is read from the upper half of this sum into the Xi register together with the result exponent.

If an overflow of the highest order coefficient bit occurs during the addition process, the result coefficient is displaced one bit and the result exponent is corrected by one count.

This instruction is intended for use in floating point calculations where rounding of opeands is not desired. This is the case in multiple precision arithmetic and in calculation involving error analysis.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

EXECUTION TIMING (continued)

CP0	Instruction issues from IPT Transmit i, j and k designators to register modules Set Xj reservation flag
CP1	Read (Xj) to floating add module FA Read complement of (Xk) to floating add module FA Compare exponents Transmit coefficients to pre-add shift register
CP2	Select smaller exponent
CP3	Shift coefficients for bit alignment Transmit coefficients to 97-bit adder
CP4	Form double precision sum
CP5	Transmit DP sum to bit position network
CP6	Determine significant bit position Transmit result to shift network
CP7	Perform 96-bit normalize shift Clear Xj reservation flag
CP8	Enter upper half of result in Xi

I 22IiJjkk I	Floating product of (Xj) times (Xk) to Xi	[22XX] [ ] [(Ax)]

This instruction multiplies two normalized floating point operands read from the Xj and Xk registers and enters the result in the Xi register. The result entered in Xi is the upper half of a double precision product.

The two operands are unpacked from floating point format (the operands are not rounded). The exponents are added to determine the exponent for the result.

The coefficients are multiplied as signed integers to form a 96-bit double precision integer product. The upper half of this product is then extracted to form the 48-bit coefficient for the result. If the double precision product has only 95 significant bits, a 1-bit normalizing shift is performed before extracting the upper half, and the exponent for the result is corrected by one count.

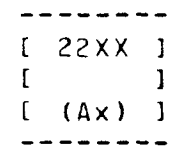
This instruction is intended for use in single and multiple precision floating point calculations. Used together with the 012 instruction, this instruction forms a 96-bit double precision product in two X registers with no loss of precision.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
X register input path is free eight clock periods after issue

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT



EXECUTION TIMING (continued)

CP0	Instruction issues from IPT Transmit i, j and k designators to register modules
CP1	Transmit (Xj) and (Xk) to floating multiply module MA Perform sign corrections Separate exponents from coefficients
CP2	Form first 16x48 product
CP3	Form second 16x48 product
CP4	Form third 16x48 product
CP5	Merge the three 16x48 products into 96-bit result register
CP8	Enter upper 48 bits in Xi

23iijjkk	Branch backward i words if (Xj) < (Xk)	[23XX] [(Bx)]

This instruction causes the current program sequence to terminate and branch backward the number of words specified by the i designator if (Xj) minus (Xk) is negative, or continue the current program address sequence if (Xj) minus (Xk) is zero or positive.

If the branch condition (Xj) < (Xk) is satisfied, the difference of the current contents of the P register and the i designator considered as a 4-bit positive integer becomes the new contents of the P register. No further instructions are issued from the current instruction word. If the instruction address stack IAS contains an address equal to the new contents of the P register, the corresponding instruction word is read to the current instruction word register CIW. If there is no address coincidence in the IAS, an instruction fetch is initiated for the new program sequence.

ISSUE CONDITIONS

Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues

EXECUTION TIMING

Execution time for this instruction is seven clock periods if the destination address is currently in the instruction address stack IAS. Minimum execution time is 20 clock periods if the destination address is not in the IAS. Delays may occur in this latter case due to storage bank conflicts or other processor conflicts in storage access control. If (Xj) < (Xk) (branch fall through), execution time is three clock periods.

BRANCH FALL THROUGH

CP00	Instruction issues from IPT
CP01	Transmit (Xj) to operand register A Transmit (Xk) to operand register B
CP02	Begin integer add Branch condition not satisfied
CP03	Next instruction may issue

EXECUTION TIMING (continued)

[23XX]
[ ]
[(Bx)]

BRANCH IN STACK

CP00 Instruction issues from IPT

CP01 Transmit (Xj) to operand register A
Transmit (Xk) to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Select P - i input to P register

CP04 Coincidence in IAS

CP05 Read IWS to CIW register

CP06 Read CIW register to IPT

CP07 Next instruction may issue

BRANCH OUT OF STACK

CP00 Instruction issues from IPT

CP01 Transmit (Xj) to operand register A
Transmit (Xk) to operand register B

CP02 Integer add front half
Branch condition satisfied

CP03 Select P - i input to P register

CP04 No coincidence in IAS
Set out of stack flag OSF

CP05 Transmit P to RA adder
Transmit reference flag to storage access control

CP06 Lower 8 bits of address to storage access control
Transmit P to IFA register
Transmit P to NSA register

CP07 Upper 12 bits of address to storage access control

CP08 Acknowledge from storage access control

CP17 Instruction word arrives at IWS

CP18 Read IWS to CIW register

CP19 Read CIW register to IPT

CP20 Next instruction may issue

C

30ijjkk | kkkkkkkk	Read data at address (Xj) + K to Xi	30XX (Cx)

This instruction reads a word of data from the object program storage field and enters that word in the Xi register. The storage address is determined by adding the sum of (Xj) and the K field from the instruction to the object program reference address.

A separate test is made to determine if the value of (Xj) + K considered as a 20-bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word and an exchange jump is made to the exchange address XA.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00 Instruction issues from IPT

CP01 Transmit (Xj) to operand register A
Transmit K to operand register B

CP02 Integer add front half

CP03 Integer add back half
Transmit integer sum to RA adder in IW module
Transmit reference flag to SA module

CP04 Lower 8 bits of address arrive at SAC

CP05 Upper 12 bits of address arrive at SAC

CP06 Acknowledge from SAC

CP15 Data word arrives at the X register

31iijjkk	Read data at address (Xj) + (Xk) to (Xi)	[31XX] [ ] [(Dx)]

This instruction reads a word of data from the object program storage field and enters that word in the Xi register. The storage address is determined by adding the sum of (Xj) plus (Xk) to the object program reference address.

A separate test is made to determine if the value of (Xj) plus (Xk) considered as a 20-bit positive integer is equal to, or greater than, the current object program field length. If this is the case, the object program is interrupted by setting the data field limit flag in the exchange parameter word. The storage reference is aborted in this case and an exchange jump is made to the exchange address XA.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

Minimum execution time for this instruction is 15 clock periods. Delays may occur in the arrival of the data word at the X register due to storage bank conflicts or other processor conflicts in storage access control.

CP00	Instruction issues from IPT
CP01	Transmit (Xj) to operand register A Transmit (Xk) to operand register B
CP02	Integer add front half
CP03	Integer add back half Transmit integer sum to RA adder in IW module Transmit reference flag to SA module
CP04	Lower 8 bits of address arrive at SAC
CP05	Upper 12 bits of address arrive at SAC
CP06	Acknowledge from SAC
CP15	Data word arrives at the X register

C

32iijjkk	kkkkkkkk	Store data at address (Xj) + K from Xi	[32XX] [ ] [(Ex)]

This instruction forms an absolute address by adding the address indicated by the sum of (Xj) and sign extended K to memory reference address RA from the processor parameter word XPW, and writes one word from the Xi register into memory at that absolute address.

If the memory field length FL is exceeded, the memory reference is aborted and an exchange jump is made to the exchange address XA in the processor parameter word XPW. If a parity error occurs in reading the old data at the indicated memory address, the error is ignored and the processor operation continues in a normal manner.

This instruction allows a processor to write data into memory from any X register.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

33ijjkk	Store data at address (Xj) + (Xk) from Xi	33XX (Fx)

This instruction forms an absolute address by adding the address indicated by the sum of (Xj) and (Xk) to memory reference address RA from the processor parameter word XPW, and writes one word from the Xi register into memory at that absolute address.

If the memory field length FL is exceeded the memory reference is aborted and an exchange jump is made to the exchange address XA in the processor parameter word XPW. If a parity error occurs in reading the old data at the indicated memory address, the error is ignored and the processor operation continues in a normal manner.

This instruction allows a processor to write data into memory from any X register.

ISSUE CONDITIONS

Xi register is free one clock period after instruction issues
Xj register is free one clock period after instruction issues
Xk register is free one clock period after instruction issues
A storage access buffer is available for this processor

EXECUTION TIMING

No execution delays possible after this instruction issues from IPT.

CP0 Instruction issues from IPT

CP1 Transmit (Xj) to operand register A
Transmit (Xk) to operand register B
Transmit (Xi) to SW module

CP2 Integer add front half

CP3 Integer add back half
Transmit integer sum to RA adder in IW module
Transmit reference flag to SA module

CP4 Lower 8 bits of address arrive at SAC

CP5 Upper 12 bits of address arrive at SAC

CP6 Acknowledge from SAC

APPENDIX

INDEX

ABBREVIATION		PAGE
( )	Indicates the contents of the register, address, etc. specified within the parenthesis	
A	Operand register A	
B	Operand register B	
CIW	Current instruction word register	
CP	Clock period	
DA	Divide approximation functional unit A	
DB	Divide approximation functional unit B	
F	4-bit instruction code	1-0
FA	Floating add functional unit A	
FB	Floating add functional unit B	
FL	Field length	

i	4-bit X register designator	1-0
I	Lower 2 or 4 bits of 6 or 8 bit instruction codes	1-0
IA	Instruction address stack module	
IAS	Instruction address stack	
IFA	Instruction fetch address	
IPF	Inter-processor interlock flag	
IPT	Instruction parcel translator	
IW	Instruction word stack module	
IWS	Instruction word stack	
j	4-bit X register designator	1-0
k	4-bit X register designator	1-0
K	20-bit program constant	1-0
MA	Floating multiply functional unit A	
MB	Floating multiply functional unit B	
MM	Memory bank module	
MTF	Monitor flag	
n	6-bit constant i, k	1-0
NSA	Next stack address	
OVF	FP interrupt flag on overflow/indefinite	

P	Program address register	
P0	Processor 0	
P1	Processor 1	
P2	Processor 2	
P3	Processor 3	
PRF	Program reference flag	
RA	Reference address	
RA	Register module, bits 00-15	
RB	Register module, bits 16-31	
RC	Register module, bits 32-47	
RD	Register module, bits 48-63	
RF	Record flag	
SA	Storage access control module A	
SAS	Storage address stack	
SB	Storage access control module B	
SW	Storage word stack module	
SWS	Storage word stack	
XAC	External access control	
Xd	X register specified by i, j, or k designators	1-0
XDW	X register data words	2-13
Xi	X register specified by i designator	1-0
Xj	X register specified by j designator	1-0
Xk	X register specified by k designator	1-0
Xs	X register specified by storage readout	
XPW	Exchange parameter word	2-13

C

QUATERNARY (di-bits)	OCTAL	DECIMAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	2	2	2
3	3	3	3
10	4	4	4
11	5	5	5
12	6	6	6
13	7	7	7
20	10	8	8
21	11	9	9
22	12	10	A
23	13	11	B
30	14	12	C
31	15	13	D
32	16	14	E
33	17	15	F
100	20	16	10
101	21	17	11
102	22	18	12
103	23	19	13
110	24	20	14
111	25	21	15
112	26	22	16
113	27	23	17
120	30	24	18
121	31	25	19
122	32	26	1A
123	33	27	1B
130	34	28	1C
131	35	29	1D
132	36	30	1E
133	37	31	1F
200	40	32	20
201	41	33	21
202	42	34	22
203	43	35	23
210	44	36	24
211	45	37	25
212	46	38	26
213	47	39	27
220	50	40	28
221	51	41	29
222	52	42	2A
223	53	43	2B

230	54	44	2C
231	55	45	2D
232	56	46	2E
233	57	47	2F
300	60	48	30
301	61	49	31
302	62	50	32
303	63	51	33
310	64	52	34
311	65	53	35
312	66	54	36
313	67	55	37
320	70	56	38
321	71	57	39
322	72	58	3A
323	73	59	3B
330	74	60	3C
331	75	61	3D
332	76	62	3E
333	77	63	3F
1000	100	64	40
1100	120	80	50
1200	140	96	60
1300	160	112	70
2000	200	128	80
2100	220	144	90
2200	240	160	A0
2300	260	176	B0
3000	300	192	C0
3100	320	208	D0
3200	340	224	E0
3300	360	240	F0
10000	400	256	100
20000	1000	512	200
30000	1400	768	300
100000	2000	1024	400
200000	4000	2048	800
300000	6000	3072	C00
1000000	10000	4096	1000
2000000	20000	8192	2000
3000000	30000	12288	3000
10000000	40000	16384	4000
20000000	100000	32768	8000
30000000	140000	49152	C000
100000000	200000	65536	10000
200000000	400000	131072	20000
300000000	600000	196608	30000
1000000000	1000000	262144	40000