CYBER
840A
850A
860A
870A

CYBER 170 State

System Description
Functional Descriptions
Operating Instructions
Instruction Descriptions
Programming

Hardware Reference

CONTROL DATA

60463560

Central Processor Instruction Index

Peripheral Processor Instruction Index

CYBER 840A 850A 860A 870A

CYBER 170 State

System Description
Functional Descriptions
Operating Instructions
Instruction Descriptions
Programming

Hardware Reference

60463560

CONTROL DATA

Revision Record

Revision	Description
A	Manual released 04-15-86.
B	Manual revised 04-30-87. ECO 48575 adds information for CYBER 870A and DMA-Enhanced CYBER 170 Channel Adapter.
C	Manual revised 12-11-87. ECO 49229 adds information for DMA-enhanced intelligent peripheral interface (IPI) channel adapter.

Address comments concerning this manual to:

Control Data Corporation
Technology and Publications Division
4201 North Lexington Avenue
St. Paul, MN 55126-6198

or use Comment Sheet in the back of this manual.

Revision letters I, O, Q, S, X, and Z are not used.

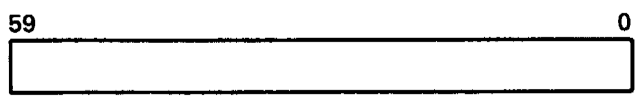
Preface

This manual contains hardware reference information for the CDC® CYBER 840A, 850A, 860A, and 870A Computer Systems.

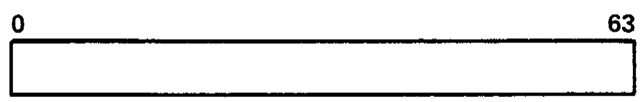
The manual describes the functional, operational, and programming characteristics of the computer system hardware. Additional hardware reference information is available in the publications listed in the related publications on the following page.

This manual is for use by customer, marketing, training, programming, and Engineering Services personnel who operate, program, and maintain the computer systems.

There are two methods used within this manual to designate bit numbers. In the majority of the manual, bits are numbered 59 through 0, reading from left to right.



However, in the context of the two-port multiplexer and maintenance registers, bits are numbered 0 through 63 from left to right.



Other manuals that are applicable to the CYBER 840A, 850A, 860A, and 870A Computer Systems but are not in the following:

Control Data Publication	Publication Number
NOS Version 2 Operator/Analyst Handbook	60459310
NOS Version 2 Systems Programmer's Instant	60459370
NOS Version 1 Operator's Guide	60457700
NOS Version 1 Systems Programmer's Instant	60457790
NOS/BE Version 1 Operator's Guide	60457380
NOS/BE Version 1 System Programmer's Reference Manual, Volume 1	60458480
NOS/BE Version 1 System Programmer's Reference Manual, Volume 2	60458490
Maintenance Register Codes Booklet	60458110
Codes Booklet	60458100
CYBER Initialization Package (CIP) User's Handbook	60457180
Hardware Operator's Guide	60463430
CDC 721 Enhanced Display Terminal (CC 634B) HRM	62950102
CYBER 845A, 850A, 860A, 870A Cooling System Hardware Maintenance Manual	60461610
CYBER 840A, 850A, 860A, 870A Power System Hardware Maintenance Manual	60461620

Control Data manuals are available through the Control Data sales office or through Control Data:

Control Data Literature Distribution Services
308 N. Dale Street
St. Paul, MN 55103

WARNING

This equipment generates, uses and can radiate radio frequency energy, and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of the FCC rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Contents

Appendix

Index

Figures

Tables

1

System Description

System Description 1

This chapter introduces the computer systems, identifies their physical and functional characteristics, and provides descriptions of major system components.

Introduction

The computer systems are large-scale, high-speed systems for both business and scientific applications. The systems include the following components.

- Central processor (CP).
- Central memory (CM).
- Input/output unit (IOU).

Physical Characteristics

The mainframe configuration for the computer system (figure 1-1) includes an interconnected three-section cabinet for the CP, CM, and IOU. System operation also requires the system console. A second CP, which is contained in an additional one-bay section is standard on an 870A and optional on an 860A. In addition to the standard IOU unit, an optional DMA (direct memory access) IOU is available with all models.

Each cabinet section contains a logic chassis with plug-in circuit boards. The CP cabinet section comprises three attached subsections, each with separate power and cooling facilities. Each cabinet section also contains an ac/dc control section with voltage margin testing facilities and dc power supplies. A stand-alone water-cooling unit(s) provides cooling for the CP subsections, CM, and IOU. For specific cooling configurations, refer to the mainframe site preparation manual listed in the preface. For additional cooling or power information, refer to the cooling system and power system manuals listed in the preface.

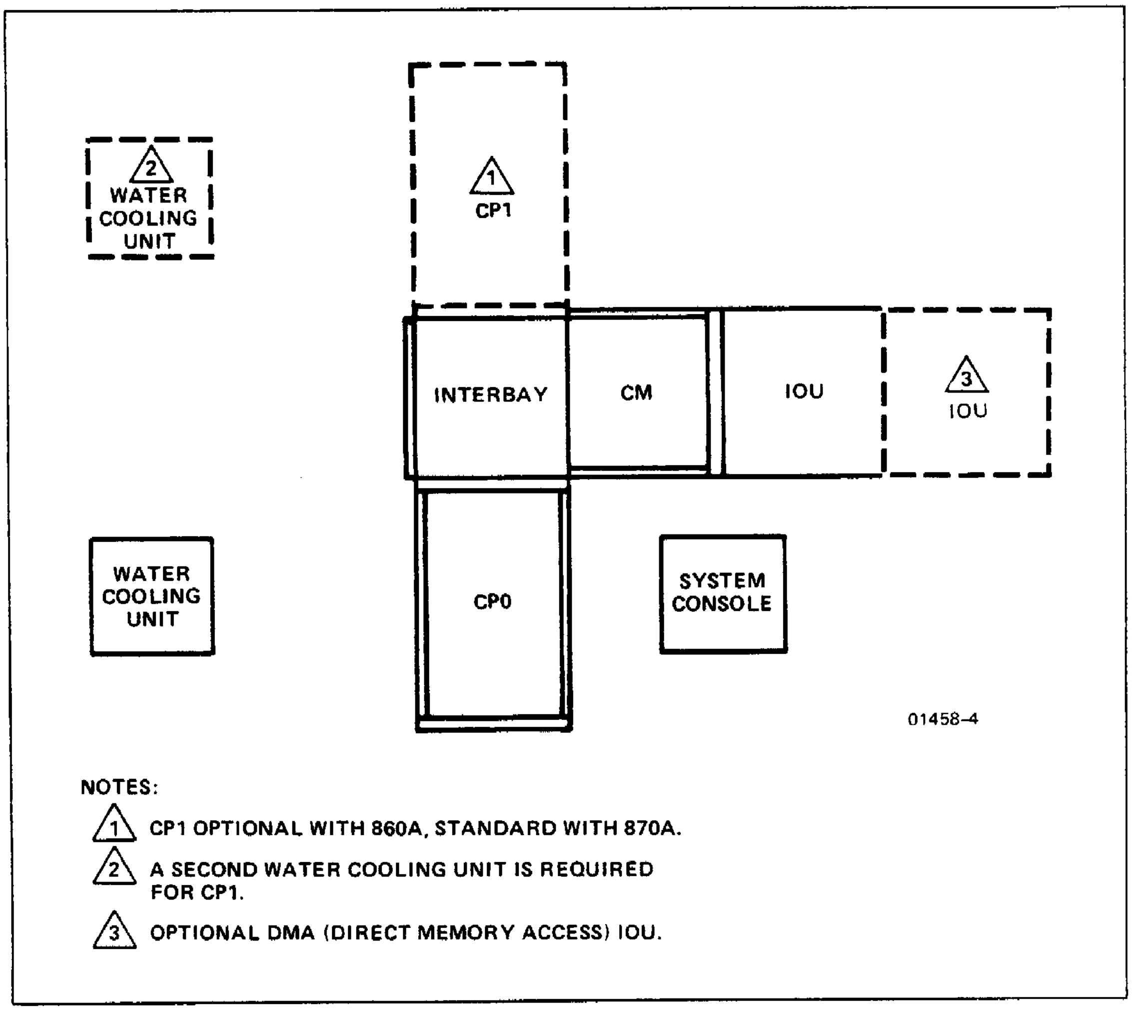


Figure 1-1. Physical Characteristics

Functional Characteristics

To achieve high computation speeds, the computer system uses emitter-coupled logic (ECL) and large-scale integration (LSI) logic. High speed is also the objective of the CP design, which is based on the assumption that both data and instructions are, in most cases, accessed from successive memory locations. Accordingly, the CP prefetches both instructions and data that are expected to be used next while the current instruction is being processed.

The semiconductor central memory is divided into eight independent banks. These banks may all be simultaneously in the process of completing read/write requests that are queued and distributed at ECL speeds. System input/output speeds are determined by the capabilities of existing external devices.

Characteristics

Central Processor

The CP has the following characteristics.

- 60-bit internal word.
- Eight 60-bit operand (X) registers.
- Eight 18-bit address (A) registers.
- Eight 18-bit index (B) registers.
- Two registers that isolate each user's central memory space (RAC, FLC).
- Two registers that isolate each user's extended memory space (RAE, FLE).
- Register exchange instructions (exchange jumps) for interrupting programs.
- Floating-point arithmetic (10-bit exponent plus sign bit, 48-bit coefficient plus sign bit). Some FP instructions use 96-bit (double-precision) coefficients.
- Integer arithmetic (60/18-bit operands).
- Character string compare/move facilities (6-bit characters).
- Packed instructions (15/30/60-bit instructions in 60-bit words).
- Synchronous internal logic.
- 64-ns clock period.
- 2048-word cache buffer memory; option available for 4096-word cache.
- Instruction and branch instruction look-ahead.
- Microcode control.
- Parity checking of all major data and address paths.
- Maintenance channel to IOU.

Central Memory

The CM has the following characteristics.

- 72-bit data word (60 data bits; 8 single-error correction, double-error detection bits; and 4 unused bits).
- 2097K words (16 Mbytes) of dynamic random access memory; options available to 167 76K words (128 Mbytes).
- Organization of eight independent banks.
- Two memory ports (located in the central processor cabinet).
- Bounds register to limit write access.
- 64-ns clock period.
- Maximum data transfer rate of one word every 32 ns.
- 464-ns read access time.
- 384-ns read/write cycle time.
- 768-ns partial write cycle time.
- Read and write data queuing capability.
- Single-error correction, double-error detection (SECDED) on stored data.
- Parity checking of all major data, address and control paths.
- Unified extended memory (UEM), which serves as extended memory within CM.

Input/Output Unit

The IOU has the following characteristics.

- Twenty peripheral processors (PPs). Each PP has 4K or 8K of independent memory (PPM) comprised of 16-bit words with the upper 4 bits equal to 0.
- Port to central memory.
- Bounds register to limit writes to central memory.
- Twenty-four 12-bit CYBER 170 channels to external devices.
- Real-time clock (channel 14_8).
- Display controller (CYBER 170 channel 10_8).
- Two-port multiplexer (channel 15_8).
- Maintenance channel (channel 17_8).
- Parity checking on all major data and address paths.
- Operating speed of 250 ns and a minor cycle of 50 ns.
- Optional concurrent input/output (CIO) PPs and direct memory access (DMA) I/O channel adapters.

Major System Component Descriptions

The major system components include:

- Central processor (CP)
- Central memory (CM)
- Input/output unit (IOU)
- System console

The remainder of the chapter provides brief descriptions of the major system components. The descriptions refer to the computer system block diagram (figure 1-2).

Central Processor (CP)

The central processor (CP) hardware (figure 1-2) consists of the following.

- Instruction section.
- Registers.
- Execution section.
- Cache memory.
- Addressing section.
- Central memory control.

The CP is isolated from the IOU and, therefore, is able to carry on computation or character manipulation unencumbered by I/O requirements.

Instruction Section

The instruction section directs the arithmetic and manipulative functions for instruction execution. The instruction section prefetches instruction words from memory and disassembles them into instructions..

Registers

Operating registers reduce storage accesses for operands used during the execution of an instruction. These registers are:

- Eight 60-bit X registers (X0 through X7), which hold operands used for computation.
- Eight 18-bit A registers (A0 through A7), which use A0 primarily for indexing and A1 through A7 for CM operand addressing.
- Eight 18-bit B registers (B0 through B7), which are primarily indexing registers to control program execution. The B0 register always contains all 0's.

Eight support registers support the operating registers during program execution. These registers are:

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register. This is a program's lower bound.
- 21-bit field length for CM (FLC) register. This is a program's upper bound.
- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21-bit reference address for UEM (RAE) register.
- 24-bit field length for UEM (FLE) register.
- 18-bit monitor address (MA) register.

The registers store data and control information, present operands to the execution section, and store results.

The operating and support registers reside in the operand issue section.

Execution Section

The execution section combines the operands to achieve the result.

Cache Memory

The cache memory consists of two sets of fast bipolar memory that are capable of storing 2048 60-bit words. Cache memory can be expanded to four sets of bipolar memory with a capacity of 4096 words. The memory addressing sections determine whether a requested word is in the cache memory. If the word is not, they read four consecutive words from central memory into the cache memory.

Addressing Section

The addressing section checks memory addresses against the CP registers RAC, FLC, RAE, and FLE to ensure isolation of user memory space.

Central Memory Control

Central memory control (CMC) is integrated within the CP. CMC controls the flow of data between CM and requesting system components.

Central Memory (CM)

The CM (figure 1-2) consists of the following items.

- Eight memory banks.
- Memory ports.
- Distributor.

The CM is a dynamic random access memory organized into eight independent banks.

A portion of CM can be reserved for use as extended memory. This portion, which is called unified extended memory (UEM), is referenced by the RAE and FLE registers. The UEM operates in either 24-bit format standard addressing mode or 30-bit format expanded addressing mode.

One memory port has a queuing buffer. The ports are located in the central processor cabinet.

The distributor resolves port conflicts and multiplexes data from ports to the storage unit. It includes the error correction code (ECC) generator, SECDED, and partial-write logic. The distributor is located in the central processor cabinet.

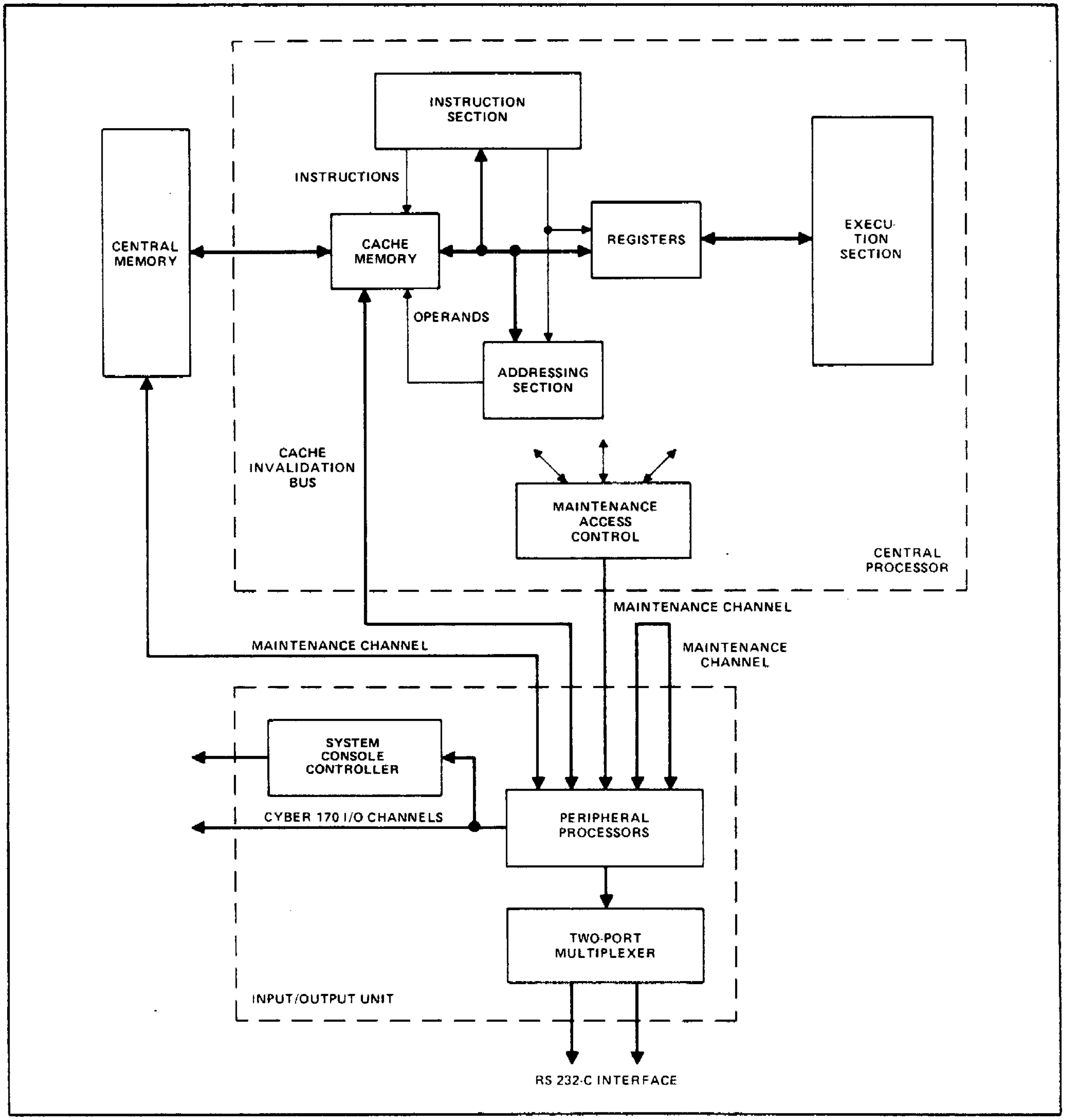


Figure 1-2. System Block Diagram

Input/Output Unit (IOU)

The input/output unit (IOU) consists of:

- Twenty logically independent, non-concurrent input/output (NIO) peripheral processors (PPs). Options are available to increase the total to 25 or 30 PPs.
- Five or ten optional logically independent, concurrent input/output (CIO) PPs and direct-memory access (DMA) channel adapters.
- Internal interface to 24 I/O channels. Options are available to increase the total to 34 channels.
- External interfaces to I/O channels:
  - 11 or 23 CYBER 170 channel interfaces.
  - Display controller interface (CYBER 170 channel 10_8).
  - Real-time clock interface (channel 14_8).
  - Two-port multiplexer interface (channel 15_8).
  - Maintenance channel interface (channel 17_8).
- Interface to central memory.
- Bounds register to limit writes to CM.

The PPs are organized in groups of five, which are called barrels. The PPs in a barrel time-share common hardware. Each PP has its own 4K or 8K independent memory and communicates with all I/O channels and with central memory.

System Console

The system console, which is required for system operation, provides a visual, alphanumeric readout for the computer. The receipt of symbol and position information from the computer enables displaying program information on a cathode-ray tube (CRT). The station also contains an alphanumeric keyboard that enables an operator to send data to the computer. The keyboard and CRT combination permits the computer operator to monitor and control system operation. Except for programming information in chapter 5, refer to the CDC 721 hardware reference manual listed in the preface for further system console information.

2

Functional Descriptions

Functional Descriptions 2

This chapter provides functional descriptions of the central processor (CP), central memory (CM), and input/output unit (IOU) as shown in the block diagrams in chapter 1. Functional descriptions for the system display station are in the CDC 721 hardware reference manual; descriptions of the water-cooling system are in the cooling system manual; both manuals are listed in the preface.

Central Processor

The CP consists of the instruction section, registers, the execution section, cache memory, the addressing section, and central memory control.

Instruction Section

The instruction section consists of logic for instruction control.

Instruction Lookahead

The instruction lookahead hardware (ILH) prefetches a maximum of 12 instructions to make the next instruction immediately available when the execution of the previous instruction is completed. This is accomplished by reading instructions from cache/CM into a series of buffer ranks.

The ILH responds to both negative and positive resolution of a conditional branch by purging the buffer ranks and reinitializing the instruction fetch unit.

When ILH detects a conditional branch, it assumes that the branch condition will be met. ILH computes the branch target address and reads instructions from cache/CM starting at the target address. If the branch is taken, the buffer ranks contain the next executable instruction words. If the branch is not taken, the hardware purges the buffer ranks and resumes prefetching at the instruction word following the unsatisfied branch instruction.

Maintenance Access Control

The maintenance access control performs initialization and maintenance operations in the CP.

Instruction Control Sequences

The instruction control section performs instruction translation and control sequences. Each control sequence obtains the necessary instruction operands from the operating registers and provides the control signals for execution. Instructions read from CM are 60-bit instruction words that are in four 15-bit groups, two 30-bit groups, or a combination of 15-bit and 30-bit groups. The 15-bit groups are termed parcels with the first parcel (parcel 0) being the highest-order 15 bits of a 60-bit CM word. Second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. The 30-bit groups contain two 15-bit parcels.

The instruction control sequences control the execution of one or more instructions of a common type. These sequences and associated instructions are briefly described in this chapter. For further information, refer to CP Instruction Descriptions in chapter 4.

Boolean Sequence

The Boolean sequence controls instructions that require bit-by-bit data manipulation. This includes both the logical and transmissive operations. The instructions requiring logical operations are:

11	Logical product (Xj) and (Xk) to Xi	BXi Xj * Xk
12	Logical sum of (Xj) and (Xk) to Xi	BXi Xj + Xk
13	Logical difference of (Xj) and (Xk) to Xi	BXi Xj - Xk
15	Logical product of (Xj) with complement of (Xk) to Xi	BXi -Xk * Xj
16	Logical sum of (Xj) with complement of (Xk) to Xi	BXi -Xk + Xj
17	Logical difference of (Xj) with complement of (Xk) to Xi	BXi -Xk - Xj

The instructions requiring transmissive operations are:

10	Transmit (Xj) to Xi	BXi Xj
14	Transmit complement of (Xk) to Xi	BXi - Xk

Shift Sequence

The shift sequence controls instructions that require shifting the 60-bit field of data within the operand word. The shift instructions are:

20	Left shift (Xi) by jk	LXi jk
21	Right shift (Xi) by jk	AXi jk
22	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj, Xk
23	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj, Xk
43	Form mask of jk bits to Xi	MXi jk

The shift sequence also controls the pack and unpack instructions. In the packed floating format, the coefficient is contained in the lower 48 bits. The sign and biased exponents are contained in the upper 12 bits. The unpack instruction obtains the packed word from the Xk register, delivers the coefficient to the Xi register, and delivers the exponent to the Bj register. The unpack and pack instructions are:

26	Unpack (Xk) to Xi and Bj	UXi Bj, Xk
27	Pack (Xk) and (Bj) to Xi	PXi Bj, Xk

The shift sequence also controls the normalize operations. The coefficient portion of the operand is repositioned, and the exponent is adjusted so that the most significant bit of the coefficient is in the highest-order bit position of the coefficient, and the exponent is decreased by the number of bit positions shifted. The normalize instructions are:

24	Normalize (Xk) to Xi and Bj	NXi Bj, Xk
25	Round normalize (Xk) to Xi and Bj	ZXi Bj, Xk

Floating-Add Sequence

The floating-add sequence controls the operations necessary to form the 48-bit floating sum with a 12-bit exponent of the floating-point sum or difference of two floating-point operands. The floating-add instructions are:

30	Floating sum of (Xj) and (Xk) to Xi	FXi Xj + Xk
31	Floating difference of (Xj) and (Xk) to Xi	FXi Xj - Xk
32	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj + Xk
33	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj - Xk
34	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj + Xk
35	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj - Xk

Floating-Multiply and Floating-Divide Sequence

The floating-multiply and floating-divide sequence controls the operation of floating-multiply, floating-divide, and population-count instructions.

The multiply instructions are:

40	Floating product of (Xj) and (Xk) to Xi	FXi Xj * Xk
41	Round floating product of (Xj) and (Xk) to Xi	RXi Xj * Xk
42	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj * Xk

The divide instructions are:

44	Floating divide (Xj) by (Xk) to Xi	FXi Xj/Xk
45	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

The population-count instruction counts the number of 1 bits in a 60-bit operand. The instruction is:

47	Population count of (Xk) to Xi	CXi Xk

Increment Sequence

The increment sequence controls the one's complement addition and subtraction of 18-bit fixed-point operands for increment instructions 50 through 77. The sequence also controls the 60-bit one's complement sum and difference values for long-add instructions 36 and 37.

The increment instructions are:

50	Set Ai to (Aj) + K	SAi Aj + K
51	Set Ai to (Bj) + K	SAi Bj + K
52	Set Ai to (Xj) + K	SAi Xj + K
53	Set Ai to (Xj) + (Bk)	SAi Xj + Bk
54	Set Ai to (Aj) + (Bk)	SAi Aj + Bk
55	Set Ai to (Aj) - (Bk)	SAi Aj - Bk
56	Set Ai to (Bj) + (Bk)	SAi Bj + Bk
57	Set Ai to (Bj) - (Bk)	SAi Bj - Bk
60	Set Bi to (Aj) + K	SBi Aj + K
61	Set Bi to (Bj) + K	SBi Bj + K
62	Set Bi to (Xj) + K	SBi Xj + K
63	Set Bi to (Xj) + (Bk)	SBi Xj + Bk
64	Set Bi to (Aj) + (Bk)	SBi Aj + Bk
65	Set Bi to (Aj) - (Bk)	SBi Aj - Bk
66	Set Bi to (Bj) + (Bk)	SBi Bj + Bk
67	Set Bi to (Bj) - (Bk)	SBi Bj - Bk
70	Set Xi to (Aj) + K	SXi Aj + K
71	Set Xi to (Bj) + K	SXi Bj + K
72	Set Xi to (Xj) + K	SXi Xj + K
73	Set Xi to (Xj) + (Bk)	SXi Xj + Bk
74	Set Xi to (Aj) + (Bk)	SXi Aj + Bk
75	Set Xi to (Aj) - (Bk)	SXi Aj - Bk
76	Set Xi to (Bj) + (Bk)	SXi Bj + Bk
77	Set Xi to (Bj) - (Bk)	SXi Bj - Bk

The long-add instructions are:

36	Integer sum of (Xj) and (Xk) to Xi	IXi Xj + Xk
37	Integer difference of (Xj) and (Xk) to Xi	IXi Xj - Xk

Compare/Move Sequence

The compare/move sequence controls data manipulation on a character basis. The compare/move instructions (also referred to as CMU instructions) are 60-bit instructions that use six support registers for source and result field CM addresses and character position offsets. The support registers load from the 60-bit instruction word. The compare/move instructions are:

464	Move indirect (Bj) + K	IM Bj + K
465	Move direct	DM
466	Compare collated	CC
467	Compare uncollated	CU

The support registers are:

- An 18-bit K1 register that specifies which relative CM address word contains the first character of the source data field.
- An 18-bit K2 register that specifies which relative CM address word contains the first character of the result field.
- A 4-bit C1 register that specifies the character position or offset of the first CM word of the source field.
- A 4-bit C2 register that specifies the character position or offset of the first CM word of the result field.
- Two 16-bit L registers (LA and LC) that specify the number of characters in the data field. The LA register is associated with K1, and the LC register is associated with K2. Instruction 464 uses 14 register bits. Instructions 465, 466, and 467 use only the lower 8 register bits.

NOTE

CMU instructions are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

CYBER 170 Exchange Sequence

The CYBER 170 exchange sequence is the method used to swap jobs in and out of execution. When a CYBER 170 exchange jump instruction occurs, the CYBER 170 exchange sequence writes the contents of the current job's CP registers (described later in this chapter) into an area of central memory called a CYBER 170 exchange package. A CYBER 170 exchange package is associated with each job. It contains sufficient information to restart a job if the job is interrupted during execution and swapped out by a CYBER 170 exchange jump. To complete the sequence, CP registers for another job are read from its CYBER 170 exchange package, and that job begins or resumes execution. For further information, refer to CYBER 170 Exchange Jump in chapter 5.

Block Copy Sequence

The block copy sequence controls the transfer of data between CM and UEM. The addition of K to the contents of Bj determines the number of words to be transferred. The starting address for CM is formed by adding either the A0 register or certain bits of the X0 register to the RAC reference address. The starting address for UEM is formed by adding certain bits of the X0 register to the RAE reference address. The block copy instructions are:

011	Block copy Bj + K words from UEM to CM	RE Bj + K
012	Block copy Bj + K words from CM to UEM	WE Bj + K

Direct Read/Write Sequence

Instructions 014 and 015 perform single-word, direct read and write operations for UEM, and instructions 660 and 670 perform single-word, direct read and write operations for central memory.

014	Read one word from UEM at (Xk + RAE) into Xj	RXj Xk
015	Write one word from Xj to UEM at (Xk + RAE)	WXj Xk
660	Read central memory at (Xk) to Xj	CRXj Xk
670	Write Xj into central memory at (Xk)	CWXj Xk

Normal Jump Sequence

The normal jump sequence controls the execution of branch instructions 02 through 07. The 02 instruction performs an unconditional jump to the Bi register address plus K. The branch address is K when i equals 0. The 02 instruction is:

02	Jump to (Bi) + K	JP Bi + K

The conditional jump instructions 03 through 07 branch to address K if the jump condition is met. These instructions are:

030	Branch to K if (Xj) = 0	ZR Xj, K
031	Branch to K if (Xj) ≠ 0	NZ Xj, K
032	Branch to K if (Xj) is positive	PL Xj, K
033	Branch to K if (Xj) is negative	NG Xj, K
034	Branch to K if (Xj) is in range	IR Xj, K
035	Branch to K if (Xj) is out of range	OR Xj, K
036	Branch to K if (Xj) is definite	DF Xj, K
037	Branch to K if (Xj) is indefinite	ID Xj, K
04	Branch to K if (Bi) = (Bj)	EQ Bi, Bj, K
05	Branch to K if (Bi) ≠ (Bj)	NE Bi, Bj, K
06	Branch to K if (Bi) ≥ (Bj)	GE Bi, Bj, K
07	Branch to K if (Bi) < (Bj)	LT Bi, Bj, K

Return Jump Sequence

The return jump sequence controls the execution of three instructions.

00	Error exit to MA or program stop	PS
010	Return jump to K	RJ K
013	Central exchange jump to (Bj) + K or monitor exchange jump to MA	XJ Bj + K

Registers

The CP contains the operating and support registers described in the following paragraphs. These registers are located in the operand issue section (figure 1-2).

The contents of these registers can be written into memory and reloaded from memory as a CYBER 170 exchange package by a single CP instruction (CYBER 170 exchange jump). Figure 2-1 shows the CYBER 170 exchange package.

The time a CYBER 170 exchange package resides in CP hardware is called an execution interval. During this interval, CP instructions can change the contents of X, A, B, and P registers. The contents of other support registers change only as a result of a CYBER 170 exchange jump. For further information, refer to CYBER 170 Exchange Jump in chapter 5.

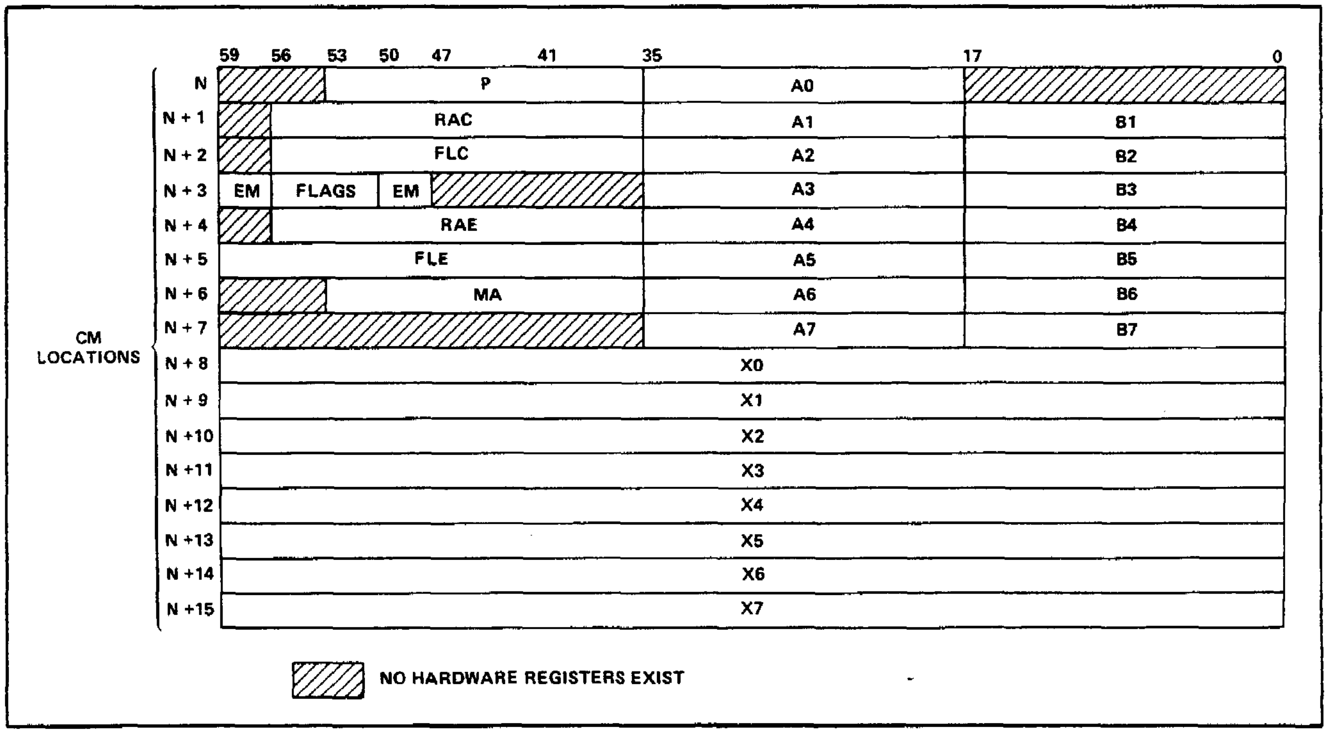


Figure 2-1. CYBER 170 Exchange Package

Operating Registers

The operating registers consist of operand (X), address (A), and index (B) registers. These registers minimize memory references for arithmetic operands and results.

X Registers

The CP contains eight 60-bit X registers (X0 through X7). The X0 register is used in the compare instructions to indicate if two fields of characters are equal. Also, the X0 register provides the relative UEM starting address in a block copy operation.

Registers X1 through X7 are primarily data-handling registers for computation. Registers X1 through X5 are used to input data from CM, and registers X6 and X7 are used to transmit data to CM.

Operands and results transfer between CM and the X registers as a result of placing CM addresses into corresponding A registers.

A Registers

The CP contains eight 18-bit A registers (A0 through A7). The A0 register serves as an intermediate register for the user's discretion. The A0 register is used in the compare collate instruction for the collate table address. Also, the A0 register provides the relative CM starting address in a block copy operation.

Registers A1 through A7 are essentially CM operand address registers associated one-for-one with the X registers. Placing a quantity into an address register (A1 through A5) causes a CM read reference to that address and transmits the CM word to the corresponding X register (X1 through X5). Similarly, placing a quantity into the A6 or A7 register causes the word in the corresponding X6 or X7 register to be written into that relative address of CM.

B Registers

The CP contains eight 18-bit B registers (B0 through B7). These registers are primarily indexing registers to control program execution. Program loop counts may also be incremented or decremented in these registers.

Program addresses may be modified on the way to an A register by adding or subtracting B register quantities. The B registers also hold shift counts for the nominal Bj shifts, the resultant exponent for the unpack, the operand exponent for the pack, and the resultant shift count from a normalize. The B0 register always contains +0, which can be used as an operand. This register cannot hold results from instructions.

Support Registers

Eight support registers assist the operating registers during programs execution. The contents of the support registers are stored in CM, and their new contents are loaded from CM during a CYBER 170 exchange sequence. With the exception of the P register, the contents of the support registers cannot be altered during the execution interval of a CYBER 170 exchange package. When the execution interval completes, the data in the support registers is sent back to CM through a CYBER 170 exchange jump.

P Register

The 18-bit program address (P) register loads from CM during the first word of a CYBER 170 exchange sequence and contains the current program execution address. The register serves as a program address counter and holds the relative CM address for each program step.

RAC Register

The 21-bit CM reference address (RAC) register loads from CM during the second word of a CYBER 170 exchange sequence. An absolute CM address forms by adding RAC to a relative address determined by the instruction. The content of the P register is added to RAC to form the program address in CM. A P-equal-to-zero condition specifies relative address 0 and, therefore, (RAC). This CM location is reserved for recording error exit conditions and should not be used to store data or instructions.

FLC Register

The 21-bit CM field length (FLC) register loads from CM during the third word of a CYBER 170 exchange sequence. The FLC register defines the size of the field of the program in execution. Relative CM addresses are compared with FLC to check that the program is not going out of its allocated memory range.

EM Register

The 6-bit exit mode (EM) register loads from CM during the fourth word of a CYBER 170 exchange sequence. The EM register holds six exit mode selection bits that control individual error conditions for a program. Selected EM register bits cause the CP to error exit when the corresponding conditions occur. Any or all of the 6 bits can be set at one time. Clear EM register bits allow the CP to continue without error processing when most of the corresponding conditions occur. Refer to the error exit tables under Error Response in chapter 5 for specific cases. The exit mode selection bits appear in the exchange package as bits 48 through 50 and bits 57 through 59. The mode selection bits and their corresponding conditions are:

Bit	Significance
48	Address out of range
49	Infinite operand
50	Indefinite operand
57	Hardware error
58	Hardware error
59	Hardware error

Flag Register

The 6-bit flag register loads from CM during the fourth word of a CYBER 170 exchange sequence. The flag register holds 6 bits that function as control flags.

Bits	Condition
51	Hardware error bit.
52	Instruction stack (lookahead) purge flag. If set, extended purging of instruction lookahead registers is enabled. For further information, refer to Instruction Lookahead Purge Control in chapter 5.
53	CMU interrupted flag. If set, one of instructions 464 through 467 has been interrupted. The information necessary to resume operation is saved.
54	Block copy flag. If set, block copy instructions (011, 012) use bits 30 through 50 of X0 rather than A0 to determine the CM address. For further information, refer to the descriptions of the block copy instructions in chapter 4.
55	Expanded addressing select flag. If set, UEM is operating in expanded addressing mode; if clear, UEM is operating in 24-bit standard addressing mode. For further information, refer to Addressing Modes under Memory Programming in chapter 5.
56	UEM enable flag. If set, UEM is available. This flag must be set to allow 011, 012, 014, and 015 instructions to access UEM.

RAE Register

The 21-bit UEM reference address (RAE) register loads from CM during the fifth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always 0. An absolute UEM address forms by adding RAE to the relative address, which is determined by the instruction.

FLE Register

The 24-bit UEM field length (FLE) register loads from CM during the sixth word of a CYBER 170 exchange sequence. The lower 6 bits of this register are always 0. The FLE register defines the size of the field in UEM for the program in execution. Relative UEM addresses are compared with FLE.

MA Register

The 18-bit monitor address (MA) register loads from CM during the seventh word of a CYBER 170 exchange sequence. The MA register contains the absolute starting address of an exchange package that is used when executing a central exchange jump (013) instruction with the CYBER 170 monitor flag clear or when honoring a monitor exchange jump to MA (262x) instruction with the CYBER 170 monitor flag clear. For further information, refer to CYBER 170 Exchange Jump in chapter 5.

Execution Section

The execution section combines the operands into results, providing additional sequencing control where necessary.

Cache Memory

Cache memory is a high-speed buffer memory that is transparent to the user. It reduces effective CM access time by eliminating unnecessary CM references. When the CP first reads CM, a block of 4 words from CM (containing the requested word) is read rapidly into cache memory. These words may be instructions or data. On subsequent reading of any of these words, CM does not have to be accessed when these words are in cache memory. Often this is the case because the same data is read more than once or because a loop of instructions is repeatedly executed. Cache memory is 2048 words or, optionally, 4096 words.

Addressing Section

An address adder calculates memory addresses for data and unconditional jump instructions.

Memory management hardware verifies that memory addresses are to access permitted memory areas. If this is the case, this hardware accesses cache memory and, if necessary, central memory.

Central Memory Control

Central memory control (CMC) provides an interface to CM for the CP and IOU. It is physically located in the CP cabinet. CMC includes:

- Ports and distributor.
- SECDED logic.
- Partial-write logic.
- Memory control logic.
- Maintenance registers.

Central Memory

The CM performs the following functions.

- The eight memory banks store from 2097K to 16 776K of 64-bit words (the leftmost 4 bits are undefined) and an 8-bit SECDED code.

- The two ports make CM accessible to the CP and every PP.

- A bounds register limits access to CM from either or both ports.

- The SECDED generators generate the SECDED code bits stored with each word. SECDED checks circuits, corrects single-bit errors, and detects double-bit errors.

- The maintenance channel interface gives a PP in the IOU access to the CM maintenance registers for system initialization, corrective action, error reporting and diagnostics, and setting the port bounds register.

Address Format

Figure 2-2 illustrates the address format for the computer system.

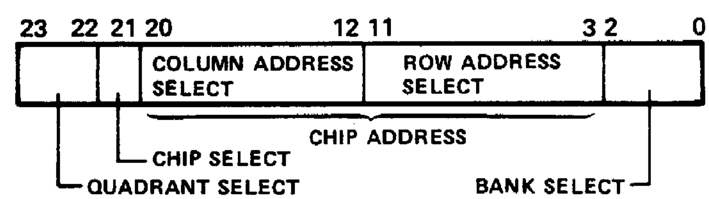


Figure 2-2. Address Format

The following list defines the address fields for figure 2-2.

- Quadrant select specifies one of four quadrants (array packs) within a bank.
- Chip select, if set, enables the row address select to the upper half (720) of the 144 chips on memory boards in all eight memory banks. If clear, chip enable enables the lower half of the 144 chips on memory boards in all eight banks.
- Chip address, which comprises column address select and row address select, specifies the address of 1 word on a chip for the selected bank and quadrant.
- Row address select specifies the row-select portion of the chip address on a chip.
- Column address select specifies the column-select portion of the chip address on a chip.
- Bank select specifies one of eight banks.

CM Access and Cycle Times

The following paragraphs list CM access and cycle times that operate on an internal clock period of 64 ns (major cycle).

The CM access time for a read operation is 320 ns (five major cycles).

One bank cycle for a read or write operation is 384 ns (six major cycles). Cycle time for a partial write (read/modify/write) is 768 ns (12 major cycles).

CM Ports and Priorities

A priority network resolves access conflicts on a rotating basis, preventing long-term lockout of any port. In case of simultaneous requests, the CP has priority.

The CM also has a refresh mechanism that may consume a maximum of 4 percent of memory time. Refresh requests have priority over port requests. Refer to table 2-1 for maximum request lockout time.

Table 2-1. Maximum Request Lockout Time in Bank Cycles

Port	Read or Write Requests
Refresh	1
Port 0	4
Port 1	5

Note: One bank cycle equals six clock periods, which equals 384 ns.

SECDED Logic

The SECDED logic corrects single-bit errors during a CM read, permitting unimpeded computer operation. The SECDED logic prepares for the error correction by generating error correction code (ECC) bits for each data word and by storing these ECC bits in CM with the data word during the CM write. Table 2-2 lists the hexadecimal codes for all the combinations of syndrome bits with the number of the data bit assigned to each code or a note categorizing the code. During a CM read, CM then performs the following SECDED sequence.

1. Read 1 CM word and generate new ECC bits for data portion of CM word.
2. Compare new ECC bits with CM word ECC bits.
3. If old and new ECC bits match, no error exists. Send data to the requesting unit.
4. If bits do not match, generate syndrome bits from the result of the ECC compare.
5. Decode syndrome bits to determine if a single- or multiple-bit failure occurred.
6. If a single-bit failure occurred, correct by inverting the failing bit in the data word. Send the corrected word to the requesting unit.
7. If a multiple-bit or other uncorrectable error occurred, send the uncorrectable error response code to the CP or the IOU. A PP in the IOU may then analyze the syndrome bits using the maintenance channel.

Table 2-2. SECDED Syndrome Codes/Corrected Bits

Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit	Code	Bit
00	⑥	20	66 ②	40	65 ②	60	③	80	64 ②	A0	③	C0	0/1 ⑤	E0	32 ①
01	71 ②	21	③	41	③	61	④	81	③	A1	④	C1	④	E1	⑤
02	70 ②	22	③	42	③	62	④	82	③	A2	④	C2	④	E2	⑤
03	6/7 ⑤	23	④	43	④	63	③	83	④	A3	③	C3	③	E3	36 ①
04	69 ②	24	③	44	③	64	④	84	③	A4	④	C4	④	E4	⑤
05	③	25	④	45	④	65	③	85	④	A5	③	C5	③	E5	34 ①
06	③	26	④	46	④	66	③	86	④	A6	③	C6	③	E6	38 ①
07	24 ①	27	⑤	47	⑤	67	30 ①	87	⑤	A7	29 ①	C7	27 ①	E7	⑤
08	68 ②	28	③	48	③	68	④	88	③	A8	④	C8	④	E8	⑤
09	③	29	④	49	④	69	③	89	④	A9	③	C9	③	E9	33 ①
0A	③	2A	④	4A	④	6A	③	8A	④	AA	③	CA	③	EA	37 ①
0B	16 ①	2B	⑤	4B	⑤	6B	22 ①	8B	⑤	AB	21 ①	CB	19 ①	EB	⑤
0C	4/5 ⑤	2C	④	4C	④	6C	③	8C	④	AC	③	CC	③	EC	35 ①
0D	8 ①	2D	⑤	4D	10 ⑤	6D	14 ①	8D	⑤	AD	13 ①	CD	11 ①	ED	⑤
0E	0 ①	2E	4 ⑤	4E	⑤	6E	6 ①	8E	⑤	AE	5 ①	CE	3 ①	EE	⑤
0F	③	2F	④	4F	④	6F	③	8F	④	AF	③	CF	③	EF	39 ①
10	67 ②	30	2/3 ⑤	50	③	70	56 ①	90	③	B0	48 ①	D0	40 ①	F0	③
11	③	31	④	51	④	71	⑤	91	④	B1	⑤	D1	⑤	F1	④
12	③	32	④	52	④	72	⑤	92	④	B2	⑤	D2	⑤	F2	④
13	④	33	③	53	③	73	60 ①	93	③	B3	52 ①	D3	44 ①	F3	③
14	③	34	④	54	④	74	⑤	94	④	B4	⑤	D4	⑤	F4	④
15	④	35	③	55	③	75	58 ①	95	③	B5	50 ①	D5	42 ①	F5	③
16	④	36	③	56	③	76	62 ①	96	③	B6	54 ①	D6	46 ①	F6	③
17	⑤	37	28 ①	57	26 ①	77	⑤	97	25 ①	B7	⑤	D7	⑤	F7	31 ①
18	③	38	④	58	④	78	⑤	98	④	B8	⑤	D8	⑤	F8	④
19	④	39	③	59	③	79	57 ①	99	③	B9	49 ①	D9	41 ①	F9	③
1A	④	3A	③	5A	③	7A	61 ①	9A	③	BA	53 ①	DA	45 ①	FA	③
1B	⑤	3B	20 ①	5B	18 ①	7B	⑤	9B	17 ①	BB	⑤	DB	⑤	FB	23 ①
1C	④	3C	③	5C	③	7C	59 ①	9C	③	BC	51 ①	DC	43 ①	FC	③
1D	⑤	3D	12 ①	5D	10 ①	7D	⑤	9D	9 ①	BD	⑤	DD	⑤	FD	15 ①
1E	⑤	3E	4 ①	5E	2 ①	7E	⑤	9E	1 ①	BE	⑤	DE	⑤	FE	7 ①
1F	④	3F	③	5F	③	7F	63 ①	9F	③	BF	55 ①	DF	47 ①	FF	③

Notes:

① Corrected single-bit error.
② Syndrome code bit failed (single code bit set).
③ Double error or multiple error (even number of code bits set).
④ Multiple error reported as a single error.
⑤ Double error or multiple error or forced double error due to a partial write parity error on one of the 2 bytes indicated.
⑥ No error detected.

CM Layout

Central memory contains an area that is reserved for special software called Virtual State software. Along with the hardware and microcode, this software handles the operations of Virtual State as described in chapter 5. Virtual State software is located at the higher end of memory. The remaining memory is available to the CYBER 170 State and may be allocated as central memory (accessible via RAC and FLC) or as unified extended memory (accessible via RAE, FLE, and the 011, 012, 014, and 015 instructions). Refer to figure 2-3.

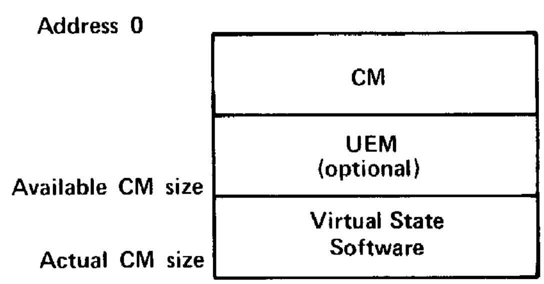


Figure 2-3. CM Layout

CM Bounds Register

The CM bounds register limits the write access to CM from specified ports. The ports are limited to the area between an upper and lower bound as specified in the CM bounds register. Bits in byte 0 specify the port(s) from which the write access is limited. The CM bounds register is set through the maintenance channel. For further information, refer to Maintenance Channel Programming in chapter 5.

Central Memory Reconfiguration

Central memory reconfiguration is a manually performed function that permits the computer operator to restructure the CM addresses so that a failing part of CM can be quickly locked out to provide a continuous block of usable CM. To accomplish CM reconfiguration, set the switches on the memory unit to manipulate the upper address bits.

When each configuration switch is set, it inverts a CM address bit. This inversion effectively moves blocks of bad memory to the highest memory block and moves blocks of good memory down, thereby, providing a sequentially addressable block of error-free memory. In case of CM malfunctions, the remaining good memory can be reconfigured so it is accessible by contiguous addresses from zero to the maximum remaining addresses. For further information, refer to chapter 3.

Input/Output Unit

The input/output unit (IOU) performs the functions required to locate, select, and initialize the external devices connected to the system. The IOU controls the transfer of data between a selected device and CM. The IOU also performs system maintenance functions.

The IOU contains the following functional areas.

- Peripheral processor (PP).
- I/O channels.
- Real-time clock.
- Two-port multiplexer.
- Maintenance channel.
- CM access.

Peripheral Processor

The basic IOU contains 20 PPs and 24 I/O channels. Each PP is a logically independent computer with its own memory. Each 5-PP group is organized into a multiplexing system that allows the PPs to share common hardware for arithmetic, logical, and I/O operations without losing independence. This multiplexing system comprises five ranks of registers, which is termed a barrel. Each rank contains information related to the instruction being executed by one PP.

Each PP can communicate with the CP by issuing a CYBER 170 exchange request to a specific CYBER 170 exchange package associated with the issuing PP. In addition, a PP can also communicate with the CP via CM read and write operations. PPs can communicate with each other over the I/O channels and through CM.

Each PP executes programs alone or with other PPs to control data transfers between external devices and CM. These programs are comprised of instructions from the IOU instruction set and respond to requests issued through CM by the operating system. The programs translate generalized operating system requests into control functions for accessing the external devices and may also perform device scheduling and optimization. The programs use PP memory as a buffer for the data transfer between external devices and CM to isolate IOU data transfer from variations in CM transfer rate.

An IOU upgrade is available which is an optional, concurrent input/output (CIO) subsystem consisting of five or ten PPs. Optional intelligent standard interface (ISI), intelligent peripheral interface (IPI), and CYBER 170 DMA (direct memory access) I/O channel adapters can be installed in the CIO.

Deadstart

A deadstart sequence allows the IOU to initialize itself. This deadstart sequence is initiated by the DEAD START switch on the system console (CC634 system console uses Control G Control R to initiate the deadstart sequence). The display includes controls for assigning any PPM to PP0. For further information, refer to chapter 3.

Barrel and Slot

The barrel consists of the R, A, P, Q, and K registers, each of which has five ranks (0 through 4). Refer to figure 2-4. Information in these registers moves from one rank to the next at a uniform 20-MHz rate, providing a multiplexed system of five PPs, each operating at a 4-MHz rate. The registers are stationary while the PPs rotate. For example, rank 4 registers contain PP0, PP1, PP2, PP3, and PP4 in succession, each of which consumes 50 ns of the total cycle time of 250 ns.

Each time data enters the slot, a portion of the instruction for that data is executed. The slot performs tasks such as arithmetic and logic operations and program address manipulation. Complete execution of an instruction may require the R, A, P, Q, and K register quantities to go more than one trip around the barrel and through the slot.

The PPM may be referenced once each time the PP passes around the barrel and through the slot. During its slot time, the PP may also communicate with CM or with any of the I/O channels.

PP Registers

The PP registers, which are discussed in the following paragraphs, are:

- R register.
- A register.
- P register.
- Q register.
- K register.

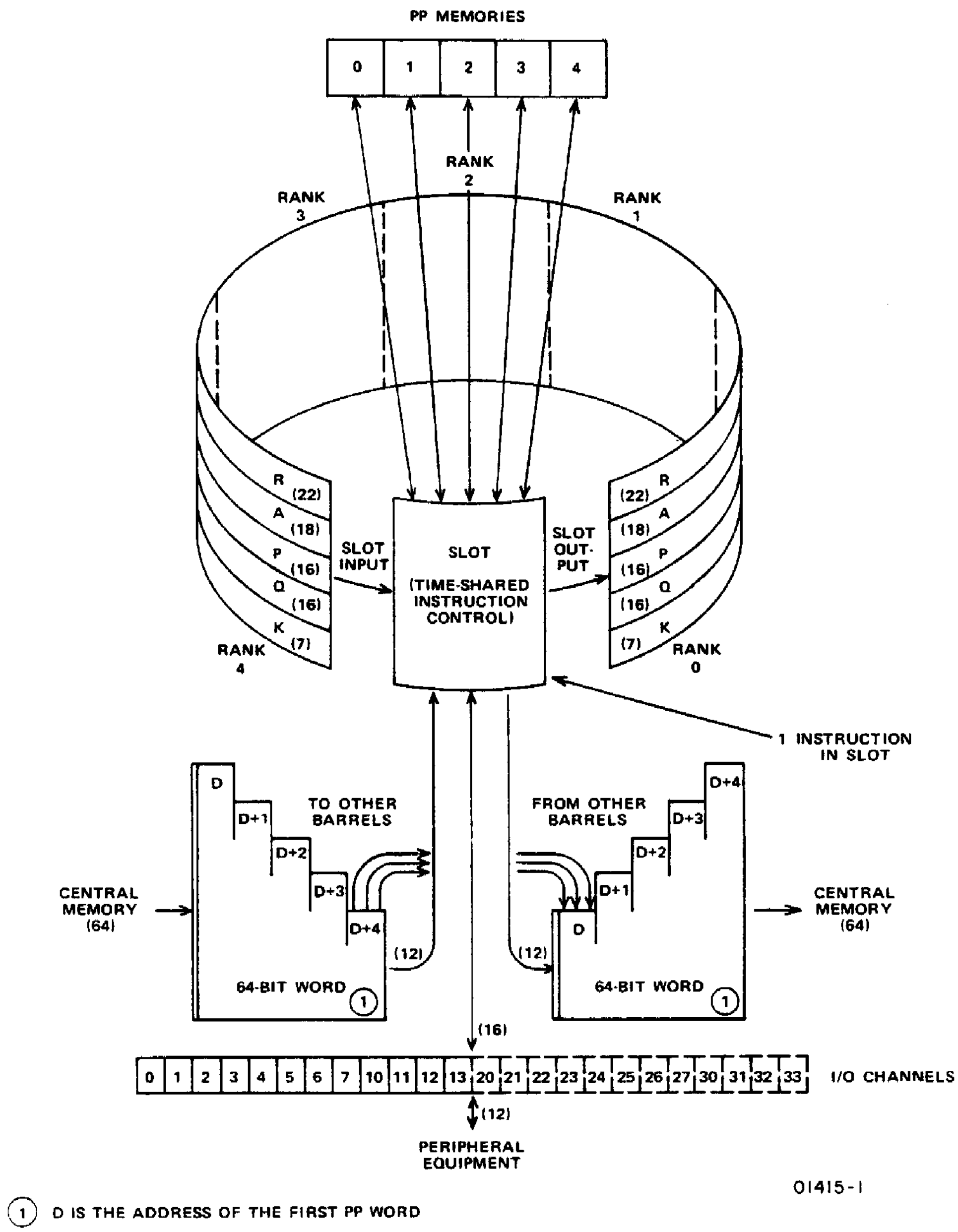


Figure 2-4. Barrel and Slot

R Register

The 22-bit R register, in conjunction with the A register, forms an absolute CM address for CM read/write instructions. When bit 17 of the A register is set, the absolute CM address is formed by appending six 0's to the lower end of the contents of the R register and adding to the result bits 0 through 16 of the contents of the A register (refer to figure 2-5).

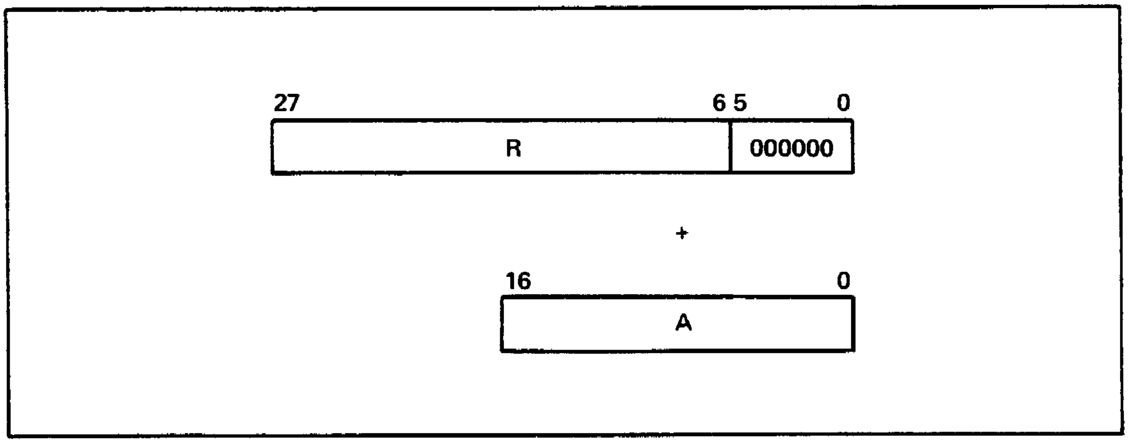


Figure 2-5. Formation of Absolute CM Address

A Register

The 18-bit A register holds one operand for arithmetic, logic, or selected I/O operations. The content of A may be an arithmetic or logical operand, CM address or part of a CM address (depending on bit 17), I/O function, I/O data word, or a word count for block I/O instructions. Various instructions operate on 6, 12, 16, or 18 bits of the A register.

When the A register is used as the CM address, parity is generated for transmission with the address to memory control. At deadstart, the A register is set to 10000 (octal). When bit 17 of the A register is clear, the A register is used as the CM address; however, when bit 17 is set, the R register is added to the A register (as described in the R register description) to obtain the absolute CM address for CM read/write instructions.

P Register

The 16-bit P register is the program address register, except during the execution of instructions 61, 63, 71, and 73. For these instructions, the P register contains the PPM address of the data transfer. At deadstart, the P register is set to 0.

Q Register

The 16-bit Q register holds data for several functions such as the address of the operand during direct addressing and indirect addressing, the peripheral address of data used during 1-word central read or write instructions, the upper 6 bits during constant mode instructions, the channel number on all I/O and channel instructions, the shift count, and the relative jump designator. At deadstart, each rank of the Q register is set to a corresponding PP number. Rank 0 is set to PP0, rank 2 is set to PP2, and so on.

K Register

The 7-bit K register is visible to the programmer through the maintenance channel only. This register holds the operation code field of an instruction for display on the IOU deadstart console and for deadstart console interrogation. When a PP is halted (idled), this register contains all 1's.

PP Numbering

PPs are numbered as follows:

Barrel	PPs
0	00 to 04
1	05 to 11 (octal)
2	20 to 24 (octal)
3	25 to 31 (octal)

The deadstart sequence is used to determine PP numbering within a barrel. The sequence assigns barrel numbers according to the IOU barrel reconfiguration parameter. During the first minor cycle after deadstart, the sequence loads a 0 into the Q register in barrel 0. This defines all the data in that rank of the barrel as belonging to PP0, and since Q is the channel selector, it assigns PP0 to channel 0. During the next minor cycle, Q loads with a 1. This defines PP1 and assigns it to channel 1. This process occurs in parallel in all barrels until the IOU assigns each rank of each barrel with a PP number and a channel number. Reassignment can be done only during a deadstart.

PP Memory

Each PP has an independent 4K or 8K word memory. Each word contains 16 data bits, with the upper 4 bits set to 0, and 6 SECDED bits. PP0 executes the deadstart program from the microprocessor RAM during the deadstart operation. PP memory 0, therefore, must be operational. A PP memory reconfiguration feature allows the user to restore IOU operation if the IOU detects a fault in the PP memory normally assigned to PP0.

To reconfigure, the operator assigns a good PP memory to PP0 and the operating system removes the failing PP memory. Computer operation can continue without the failing PP memory, and repairs can be made during scheduled maintenance. The system must be deadstarted to reconfigure PPMs.

I/O Channels

The I/O channels are composed of:

- An internal interface that allows common hardware and software to control the external devices, and

- An external interface that allows the IOU to communicate with the external devices using 12-bit data channels. The internal interface can transfer 16-bit data words between two PPs or between a PP and an external device at a maximum rate of 1 word every 250 ns.

This rate can be sustained for the maximum practical channel transfer (4096 words). During transfers between PPs, if the PPs are in the slot at the same time, the transfer rate is 500 ns.

Any PP can access any of the CYBER 170 bidirectional I/O channels. All PPs communicate with external devices through the independent I/O channels. Each channel may be connected to one or more pieces of external equipment, but only one piece of equipment can use a channel at one time. All channels can be active simultaneously. Available channels are:

- Twenty-four CYBER 170 compatible I/O channels available with a maximum data transfer rate of 3 Mbytes/second.

- An optional, DMA-enhanced, intelligent standard interface (ISI) channel adapter, intelligent peripheral interface (IPI) channel adapter or CYBER 170 channel adapter that can be installed in any one of ten channel locations in the CIO cabinet. The adapters transfer data between the ISI or CYBER 170 channel and PP memory using standard I/O instructions. They also support DMA transfer in which data goes directly between CM and an external device without going through the PP. There are two types of CYBER 170 DMA transfers, fast and normal. Fast transfers are used with the Extended Semiconductor Memory-II (ESM-II), and normal transfers are used with other CYBER 170 external devices.

The display station controller (DSC) is attached to CYBER 170 channel 10_8. The DSC is the IOU interface between the PPs and the system console, servicing both the keyboard and the cathode-ray tube (CRT). It transmits function words and digital symbol size/position data to the system console, and receives digital character codes from the keyboard. It also receives digital symbol codes from the PPs and converts these to analog signals to the CRT.

Real-Time Clock

The real-time clock is a 12-bit, free-running counter, incrementing at a 1-MHz rate. It is permanently attached to channel 14_8. This channel may be read at any time because its active and full flags are always set.

Two-Port Multiplexer

The two-port multiplexer provides communication capability between a PP and two attached terminals. One port is reserved for maintenance purposes, and the other port is reserved for future use. The two-port multiplexer is permanently attached to channel 15_8.

Maintenance Channel

The maintenance channel is used for initialization of the CP and CM maintenance registers and monitoring of error status.

The maintenance channel consists of the maintenance channel interface on channel 17_8, a maintenance access control in each system element, and a set of interconnecting cables.

Central Memory Access

Any PP can access CM. During a write from the IOU to CM, the IOU assembles five successive 12-bit PP words into a 64-bit CM word with the leftmost 4 bits undefined. During a CM read, the IOU disassembles the rightmost 60 bits of the 64-bit CM word into five PP words. To find the CM address, a PP reads the A register. If bit 17 of the A register is clear, the PP uses the contents of the A register for the CM address. If bit 17 of the A register is set, the PP adds the relocation address from the R register to the A register to form the CM address.

A maximum of 20 PPs in various stages of assembly/disassembly can simultaneously read CM words, and five PPs can write CM words.

3

Operating Instructions

Operating Instructions 3

This chapter describes mainframe controls and indicators and the operating procedures that are hardware-dependent. Software-dependent procedures are in system software reference manuals listed in the preface.

Controls and Indicators

This chapter describes IOU deadstart controls and indicators and CM configuration switches that the system operator uses. Other controls that maintenance personnel use are described in the hardware operator's guide and the power distribution and warning system, the cooling system, and the CDC 721 manuals, which are listed in the preface.

Deadstart Displays/Controls

Pressing the deadstart pushbutton on the CC545 system console or pressing the CTRL G and CTRL R keys on the CC634B system console initiates deadstart and an initial deadstart display appears on the system control screen. The display is created by an independent microcomputer in the mainframe and does not rely on any program being operational in the PPs. The initial deadstart display is used to select a 16-word deadstart program for PP0 and to initiate the deadstart sequence for PP0. The display is also used to reconfigure PPMs and barrels, and to display error status and maintenance information.

Figure 3-1 shows the format of the deadstart options display, and figure 3-2 shows the deadstart display. Table 3-1 describes the two operator-selectable options and table 3-2 describes the operator entries and functions for the deadstart display. Other deadstart displays are available for maintenance use. Refer to the CYBER Instruction Package (CIP) listed in the preface for additional information.

```
                  DEADSTART OPTIONS

         S        SYSTEM LOAD OPTIONS
         M        MAINTENANCE OPTIONS

      (CR)        SYSTEM LOAD OPTIONS



                  PROGRAM X SELECTED
```

Figure 3-1. Deadstart Options Display

```
                DEADSTART - REV. 01

XX YYYYYY=CHANGE DS PRG        PPM CONF = 00
XX+YYYYYY=CHANGE DS PRG INC    NIO BRL CONF = 0
    S=SHORT DS                 DLY LOOP = 0
    L=LONG DS                  LDS ADDR = 6000
    H=HELP                     CLK FREQ = NORMAL
                               NIO MEM SIZE = 4K

    PROGRAM 1

01  001402
02  007306
03  000017
04  007546
05  007706
06  000120
07  007406
10  007106
11  007301
12  000710
13  000000
14  000000
15  000000
16  000000
17  000000
20  007112
```

Figure 3-2. Initial Deadstart Display

Table 3-1. Deadstart Options Display

Option	Description
S	Selects a short deadstart sequence using the deadstart program identified at the bottom of the display. Upon completion of the deadstart sequence a display for loading system software appears.
M	Causes the deadstart display to appear on the screen.

Table 3-2. Deadstart Display Operator Entries and Functions

Operator Entry	Function
xx yyyyyy	Enters a single word in the deadstart program at xx to a new value yyyyyy (octal).
xx+yyyyyy	Changes words in the deadstart program in sequence starting at xx.
S	Selects a short deadstart sequence.
L	Selects a long deadstart sequence.
H	Brings up a display that lists and explains all available commands. Refer to the Hardware Operator's Guide for detailed information about these commands.

Central Memory Controls

The CM contains six two-position configuration switches (figure 3-3). These switches are located along the address interface pak switch in the A section of the memory cabinet.

The switches are used to eliminate CM sections with malfunctions. Each switch, SW0 through SW5, inverts the corresponding CM address bit (37 through 42). The inversion effectively moves blocks of bad memory to the highest memory block and moves blocks of good memory down, thereby providing a sequentially addressable block of error-free memory. Refer to table 3-3.

In case of CM malfunctions, the remaining good memory can be reconfigured so it is accessible by contiguous addresses from zero to the maximum remaining address. This is accomplished by setting configuration switches (figure 3-3) as listed in table 3-3. Refer to the hardware operator's guide listed in the preface for further information.

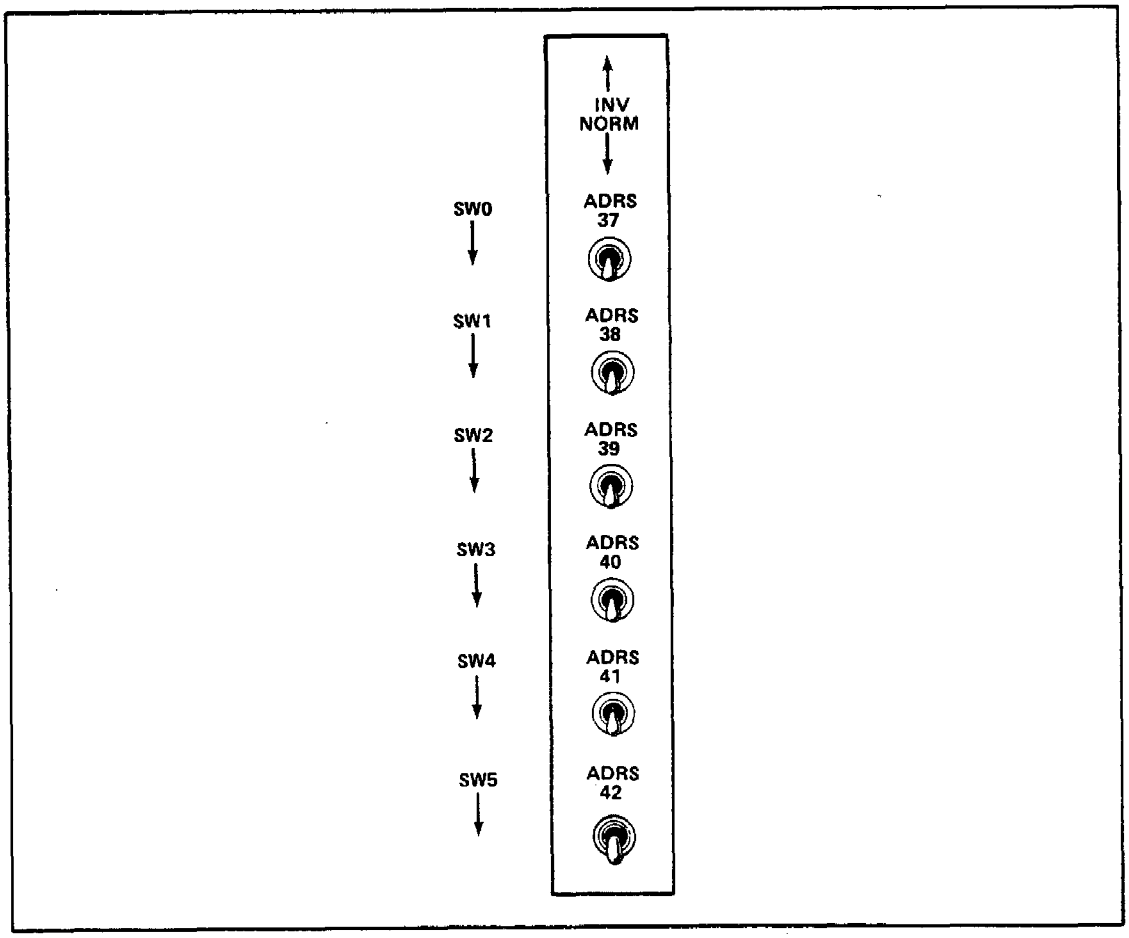


Figure 3-3. CM Configuration Switches

Table 3-3. Central Memory Reconfiguration

Original CM			Reconfigured CM					
			Reconfiguration Settings					
Size Words (MB)	Address Range	Error-Free Size	SW0 ADRS 37	SW1 ADRS 38	SW2 ADRS 39	SW3 ADRS 40	SW4 ADRS 41	SW5 ADRS 42
2097 K (16 MB)	0-7 777 777	1049 K (8 MB)	D	D	D	U	D	D
4195 K (32 MB)	0-17 777 777	2097 K (16 MB)	D	D	U	D	D	D
8390 K (64 MB)	0-37 777 777	4195 K (32 MB)	D	U	D	D	D	D
16780 K (128 MB)	0-77 777 777	8390 K (64 MB)	U	D	D	D	D	D

Notes:

1. CM remaining can be further reconfigured to obtain larger contiguous blocks of error-free memory by setting additional configuration switches. See examples shown in figure 3-4.

2. U equals up; D equals down. Normal setting of all switches is down.

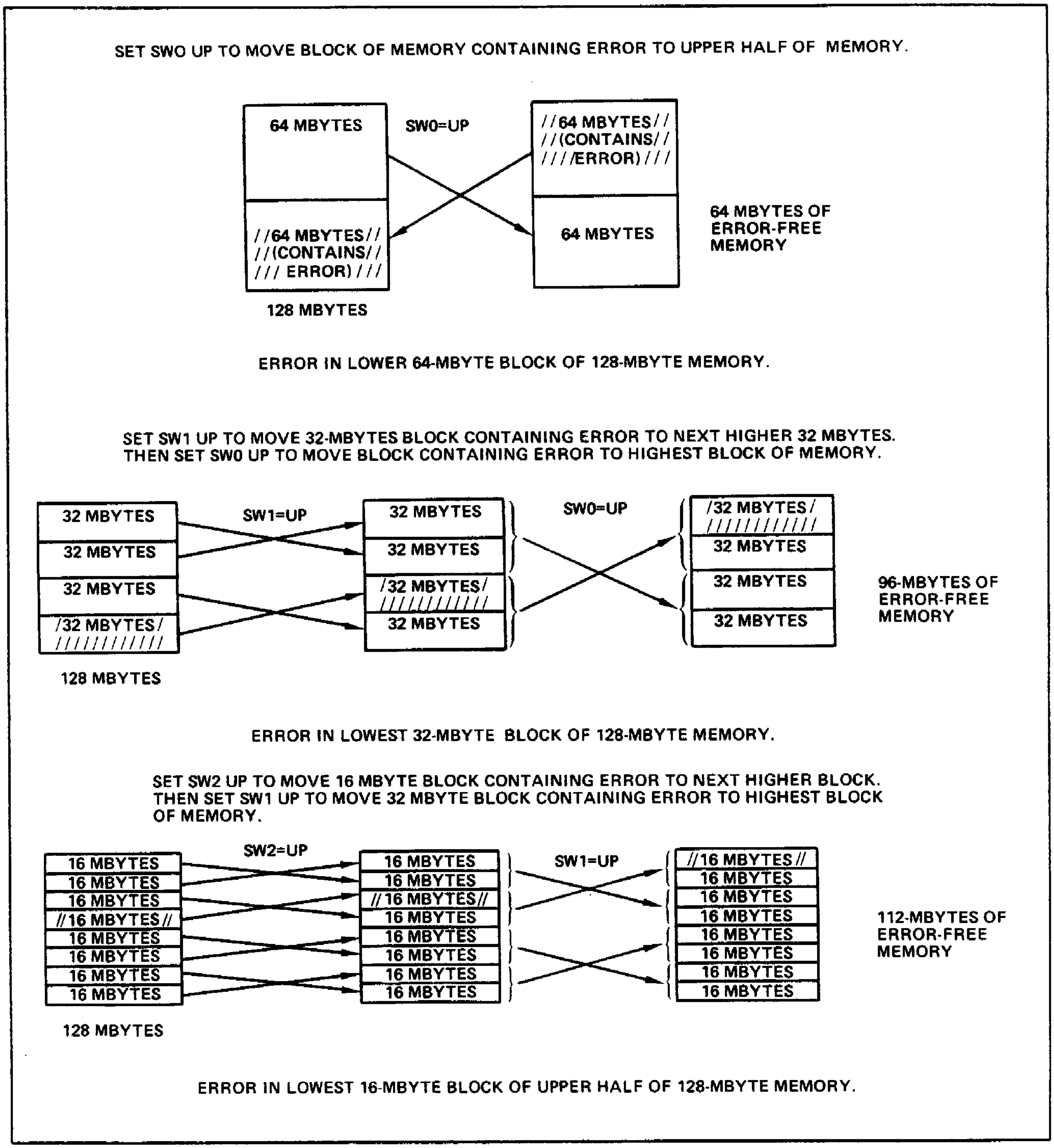


Figure 3-4. Reconfiguration Examples

Power-On and Power-Off Procedures

In case of an emergency, use the system EMERGENCY OFF switch. The power-on and power-off procedures are described in the hardware operator's guide listed in the preface.

CAUTION

Improper application or removal of power may damage system circuits and/or air-conditioning system. Power must be turned on/off by designated personnel only, except for the system EMERGENCY OFF switch. Use only for extreme emergency and not for normal shutdown.

Operating Procedures

Refer to the hardware operator's guide. The system is initialized by setting its deadstart display control parameters and then by running either a long or short deadstart sequence (defined later in this chapter). After initialization, the keyboard is used to instruct the system further under program control.

Control Checks

Before activating a long or short deadstart sequence, check the deadstart display parameters against their intended use. The normal settings of these parameters are:

Parameter	Value
PPM CONF	00
NIO BRL CONF	0
LDS ADDR	6000
Error messages	None

Deadstart Sequences

In response to a keyboard command (L or S) to the deadstart display, the IOU performs a deadstart sequence. Depending on the command (L or S), either the long or the short deadstart sequence is performed. The short deadstart sequence is used when hardware integrity verification is not required. The long deadstart sequence performs all the tasks performed by the short deadstart sequence and some additional tasks. The main additional task is the running of a diagnostic program, from a read-only memory (ROM) in the IOU on logical PP0. The diagnostic program takes approximately 15 seconds to run.

Both deadstart sequences begin with a master clear, which sets up all PPs except logical PP0, for a 4096-word block input starting at PP location 0. The input into each PP is from the channel with the same number as the logical number of the PP concerned. The master clear also resets all external devices and sets maintenance channel connect code bit 52. The individual registers are set as follows:

Register	Initialization	Description
K	007100_8	Instruction display.
P	007777_8	Causes block input to start from location 0.
A	$10,000_8$	Count of 4096 words.
Q	0, 1, 2...	I/O channel numbers (PP0: 0, PP1: 1, and so on).

All registers in both barrels are set to these values, except the registers of PP0.

If the long deadstart sequence is being performed, hardware clears location 7777_8 in all PP memories and sets the P register of PP0 to the value indicated by the parameter LDS ADDR = XXXX (normally 6000_8). PP0 starts performing a test program from a read-only memory in IOU. Hardware errors cause the LDS program to hang before completion. In the absence of errors, execution proceeds until the test program reaches location 7776_8. When this happens, the unique part of the long deadstart sequence ends with a master clear.

Next, both deadstart sequences clear PP0 location 0, write the deadstart program on the display into PP0 memory locations 1 to 20_8, and clear PP0 location 21_8. PP0 then starts executing the program entered from the deadstart display, which is normally a bootstrap program to input more data from an assigned external device.

The short deadstart sequence does not disturb PP memory other than PP0 locations 0 to 21_8. Both deadstart sequences leave all PPs, except PP0, waiting for a block input or for action through the maintenance channel. After the block input is completed, each PP starts executing the program entered from whatever address was entered into location 0 of that PP.

IOU Reconfiguration

The logical PP numbers and hardware are assigned to physical PPs circularly from the settings of IOU deadstart display PPM CONF and BRL CONF parameters, specifying which physical barrel and PPM is PP0. Maximum values for these parameters depend on the number of PPs installed (table 3-4). Illegal values entered in RB X and RP XX commands are rejected by the deadstart display and cause error messages to appear on the screen (refer to the hardware operator's guide). Reconfiguration is discussed in detail in the hardware operator's guide. Tables 3-5 and 3-6 show allowable values for the PPM CONF and BRL CONF parameters and reconfiguration examples.

Table 3-4. Barrel Numbering Table

		Logical PPs in Physical Barrel With BARREL RECONFIGURATION Switch Values			
Barrels Installed	Physical Barrel	0	1	2	3
Four Barrels (20 PPs)	0	0-4	25-31	20-24	5-11
	1	5-11	0-4	25-31	20-24
	2	20-24	5-11	0-4	25-31
	3	25-31	20-24	5-11	0-4

Table 3-5. PP and Barrel Reconfiguration Example, RP=0

No. of PPs	Physical PPMs in Each Barrel	Logical PP RB=0				Logical PP RB=1				Logical PP RB=2				Logical PP RB=3			
		BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3
20	00	00	05	20	25	25	00	05	20	20	25	00	05	05	20	25	00
	01	01	06	21	26	26	01	06	21	21	26	01	06	06	21	26	01
	02	02	07	22	27	27	02	07	22	22	27	02	07	07	22	27	02
	03	03	10	23	30	30	03	10	23	23	30	03	10	10	23	30	03
	04	04	11	24	31	31	04	11	24	24	31	04	11	11	24	31	04

Notes:

1. RP = PP Configuration.
2. RB = NIO Barrel Configuration only.
3. BAR 0-3 are the physical barrels.

Table 3-6. PP and Barrel Reconfiguration Example, RP=2

No. of PPs	Physical PPMs in Each Barrel	Logical PP RB=0				Logical PP RB=1				Logical PP RB=2				Logical PP RB=3			
		BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3	BAR0	BAR1	BAR2	BAR3
20	00	03	10	23	30	30	03	10	23	23	30	03	10	10	23	30	03
	01	04	11	24	31	31	04	11	24	24	31	04	11	11	24	31	04
	02	00	05	20	25	25	00	05	20	20	25	00	05	05	20	25	00
	03	01	06	21	26	26	01	06	21	21	26	01	06	06	21	26	01
	04	02	07	22	27	27	02	07	22	22	27	02	07	07	22	27	02

Notes:

1. RP = PP Configuration.
2. RB = IOU Barrel Configuration only.
3. BAR 0-3 are the physical barrels.

4

Instruction Descriptions

Instruction Descriptions 4

This chapter contains the CYBER 170 State CP instruction descriptions and PP instruction descriptions.

CP Instruction Formats

NOTE

CYBER 170 CP instructions use the rightmost 60 bits in the 64-bit word. The leftmost 4 bits are undefined. For these instructions, the most-significant bit is bit 59 and the least-significant bit is bit 0.

Program instruction words are divided into 15-bit fields called parcels. The first parcel (parcel 0) is the highest-order 15 bits of the 60-bit word. The second, third, and fourth parcels (parcels 1, 2, and 3) follow in order. Figure 4-1 shows possible parcel arrangements for instructions within a program instruction word.

An instruction may occupy one, two, or four parcels. This arrangement depends on the instruction format. When an instruction occupies two parcels, it must occupy two parcels within the same program word. A program word may be filled with a one-parcel pass instruction or an instruction acting as a two-parcel pass instruction. These instructions are used to fill a program word when necessary to place a particular instruction in the first parcel of a program word or to avoid starting a two-parcel instruction in the fourth parcel of a program word. Pass instructions may also be used for branch entry points because a branch instruction destination address must begin with a new word. One-parcel pass instructions are 460xx through 463xx. Instructions 60xxx through 62xxx may be used as two-parcel pass instructions by setting the i instruction designator to 0. Refer to table 4-1 for CP instruction designators.

CP instructions 011 and 012 have special properties. They are 60-bit double instructions that must start at parcel 0. The programmer has the option of providing a branch instruction at parcels 2 and 3 in the same instruction word (to an error-handling software routine) or filling this space with pass instructions. Refer to instructions 011 and 012.

Instructions 013 and 464 through 467 are 60-bit instructions which must start at parcel 0. They ignore any information in parcels 2 and 3; however, these parcels are normally set to all 0's.

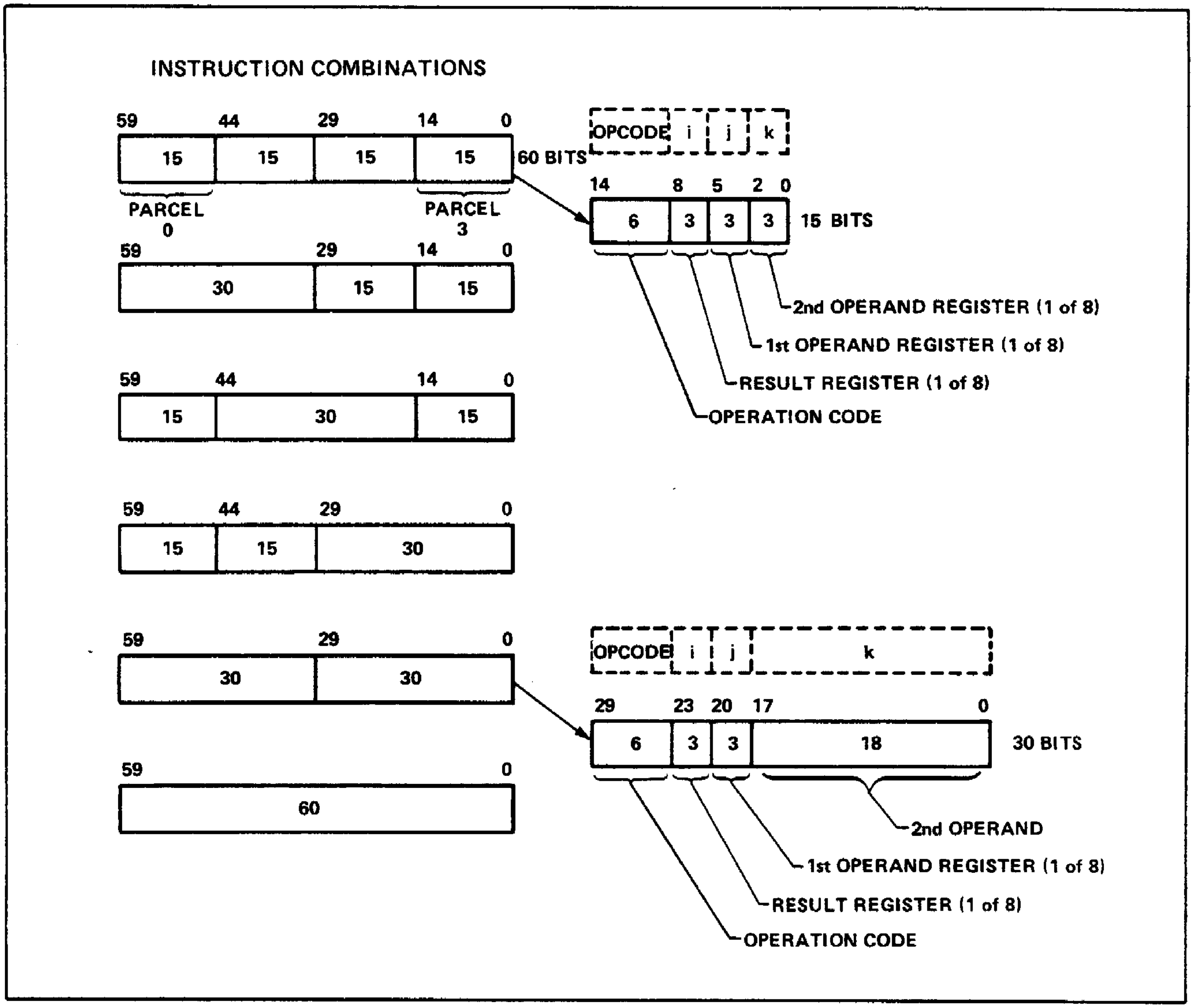


Figure 4-1. CP Instruction Parcel Arrangement

Instruction Description Nomenclature

The instruction descriptions in this chapter use the following instruction designators.

Designator	Description
Opcode	6-bit/9-bit field specifying instruction operation code.
i,j,k	3-bit code specifying one of eight registers.
jk	6-bit code specifying amount of shift or mask.
K	18-bit operand or addresss.
x	Unused designator.
A	One of eight 18-bit address registers.
B	One of eight 18-bit index registers; B0 is fixed and equal to 0.
X	One of eight 60-bit operand registers.
()	Content of the word at a central memory address.
C1†	Offset (character address) of the first character in the first word of the source field.
C2†	Character address of the first character in the first word of the result field.
K1†	18-bit address indicating the central memory location of the first (leftmost) character of the source field.
K2†	18-bit address indicating the central memory location of the first (leftmost) character of the result field.
LL†	Lower 4 bits of the field length (character count) for a move or compare instruction; used with LU to specify field length.
LU†	Upper 9 bits of the field length; (character count) for indirect move instruction or the upper 3 bits for direct instructions; used with LL to specify field length.

† Applicable to compare/move instructions only.

CP Operating Modes

The CP executes instructions in CYBER 170 job mode, CYBER 170 monitor mode, and executive state. Changes between CYBER 170 job mode and CYBER 170 monitor mode are caused by CYBER 170 exchange jumps (CP instruction 013 and PP instructions 2600, 2610, and 2620). A hardware flag called the CYBER 170 monitor flag (MF) indicates whether the CP is in CYBER 170 job mode (flag is clear) or in CYBER 170 monitor mode (flag is set).

The executive state is invisible to the applications programmer. It sets up the CYBER 170 environment during initialization, executes certain instructions, and handles hardware-detected error conditions. Hardware-caused exchanges are called error exits. Most of these can be enabled or disabled by setting or clearing bits in the CYBER 170 exchange package. For further information on CP operating modes, refer to CYBER 170 Exchange Jump, Executive State, and Error Response in chapter 5.

CP Instruction Descriptions

The CP general instructions are divided into 16 subgroups as follows:

- Integer Arithmetic
- Branch
- Block Copy
- Shift
- Logical
- Floating Point
- Jump
- Exchange/Jump
- Compare/Move
- Set
- Normalize
- Pass
- Illegal Instruction
- Mask
- Pop Count
- Read Free Running Counter

CP Integer Arithmetic Instructions

The integer arithmetic instructions (table 4-1) perform integer arithmetic on signed two's complement words or half words in Xk or XkR. The sign bit is bit 0 for full-word integers or bit 32 for half-word integers.

Table 4-1. CP Integer Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
27	ijk	Pack (Xk) and (Bj) to Xi	PXi Bj Xk
26	ijk	Unpack (Xk) to Xi and Bj	UXi Bj Xk
36	ijk	Integer sum of (Xj) and (Xk) to Xi	IXi Xj+Xk
37	ijk	Integer difference of (Xj) and (Xk) to Xi	IXi Xj-Xk

Integer Pack/Unpack

27ijk Pack (Xk) and (Bj) to Xi PXi Bj, Xk

14 … 9	8 … 6	5 … 3	2 … 0
27	i	j	k

This instruction reads the contents of Xk and Bj, packs them into a single word in floating-point format, and delivers this result to Xi. The coefficient for the value in Xi is obtained from the content of Xk, which is treated as a signed integer. The exponent for the value in Xi is obtained from the content of Bj, which is treated as a signed integer.

The lowest-order 48 bits in Xi are copied directly from the lowest-order 48 bits in Xk. The sign bit in Xi is copied directly from the sign bit in Xk. The exponent field in Xi is derived from the value in Bj by extracting the lowest-order 11 bits in Bj and modifying this quantity for exponent bias and coefficient sign.

Four sample sets of operands and packed results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk) = 0000 4500 3333 2000 0077

(Bj) = 00 0034

(Xi) = 2034 4500 3333 2000 0077

(Xk) = 0000 4500 3333 2000 0077

(Bj) = 77 7743

(Xi) = 1743 4500 3333 2000 0077

(Xk) = 7777 3277 4444 5777 7700

(Bj) = 00 0034

(Xi) = 5743 3277 4444 5777 7700

(Xk) = 7777 3277 4444 5777 7700

(Bj) = 77 7743

(Xi) = 6034 3277 4444 5777 7700

This instruction converts a number in fixed-point format to floating-point format. For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

26ijk   Unpack (Xk) to Xi and Bj   UXi Bj, Xk

14 9	8 6	5 3	2 0
26	i	j	k

This instruction reads one operand from Xk, unpacks this word from floating-point format, and delivers the coefficient and exponents to Xi and Bj, respectively. The 60-bit word delivered to Xi consists of the lowest 48 bits unaltered from the original operand plus the upper 12 bits, each equal to the original sign bit. This is a signed integer equal to the value of the coefficient in the original operand. The 18-bit quantity delivered to Bj is a signed integer equal to the value of the exponent in the original operand. The 11-bit exponent field in the operand is altered to remove the bias and then sign-extended to fill out the 18-bit quantity. The sign of the coefficient is removed in this process.

Four sample sets of operands and unpacked results are listed in octal notation to illustrate the operation performed. These examples contain the four combinations of coefficient sign and exponent sign.

(Xk) = 2034 4500 3333 2000 0077

(Xi) = 0000 4500 3333 2000 0077

(Bj) = 00 0034

(Xk) = 1743 4500 3333 2000 0077

(Xi) = 0000 4500 3333 2000 0077

(Bj) = 77 7743

(Xk) = 5743 3277 4444 5777 7700

(Xi) = 7777 3277 4444 5777 7700

(Bj) = 00 0034

(Xk) = 6034 3277 4444 5777 7700

(Xi) = 7777 3277 4444 5777 7700

(Bj) = 77 7743

This instruction converts a number from floating-point format to fixed-point format. For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

36ijk   Integer sum of (Xj) and (Xk) to Xi   IXi Xj + Xk

14 9	8 6	5 3	2 0
36	i	j	k

This instruction reads operands from two X registers, operates on them to form a 60-bit integer sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The resulting integer sum is delivered to Xi. Overflow is not detected.

This instruction adds integers too large for handling by 50 through 77 instructions. The instruction also merges and compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in chapter 5.

37ijk   Integer difference of (Xj) and (Xk) to Xi   IXi Xj - Xk

14 9	8 6	5 3	2 0
37	i	j	k

This instruction reads operands from two X registers, operates on them to form a 60-bit integer difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are signed integers. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi. Overflow is not detected.

This instruction subtracts integers too large for handling by 50 through 77 instructions. The instruction also compares data fields during data processing.

For further information, refer to Integer Arithmetic under CP Programming in chapter 5.

CP Branch Instructions

The branch instructions (table 4-2) consist of both conditional and unconditional branch instructions. Each conditional branch instruction compares the contents of two general registers to determine whether a normal or a branch exit is taken.

Table 4-2. CP Branch Instructions

Opcode	Format	Instruction	Mnemonic
030	jK	Branch to K if (Xj) = 0	ZR
031	jK	Branch to K if (Xj) $\neq$ 0	NZ
032	jK	Branch to K if (Xj) is positive	PL
033	jK	Branch to K if (Xj) is negative	NG
034	jK	Branch to K if (Xj) is in range	IR
035	jK	Branch to K if (Xj) is out of range	OR
036	jK	Branch to K if (Xj) is definite	DF
037	jK	Branch to K if (Xj) is indefinite	ID
04i	jK	Branch to K if (Bi) = (Bj)	EQ
05i	jK	Branch to K if (Bi) $\neq$ (Bj)	NE
06i	jK	Branch to K if (Bi) $\geq$ (Bj)	GE
07i	jK	Branch to K if (Bi) $<$ (Bj)	LT

Branch

030jK Branch to K if (Xj) = 0 ZR Xj, K

29 … 21	20 … 18	17 … 0
030	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if: (Xj) = 0000 0000 0000 0000 0000 (positive zero)
              (Xj) = 7777 7777 7777 7777 7777 (negative zero)
```

This instruction branches on a zero result from either a fixed-point or a floating-point operation.

031jK Branch to K if (Xj) ≠ 0 NZ Xj, K

29 … 21	20 … 18	17 … 0
031	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if: (Xj) = 0000 0000 0000 0000 0000 (positive zero)
             (Xj) = 7777 7777 7777 7777 7777 (negative zero)
```

This instruction branches on a nonzero result from either a fixed-point or a floating-point operation.

032jK   Branch to K if (Xj) is Positive   PL Xj, K

29 – 21	20 – 18	17 – 0
032	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

Jump to K if: Bit 59 of Xj = 0 (positive)
Continue if: Bit 59 of Xj = 1 (negative)

This instruction branches on a positive result from either a fixed-point or a floating-point operation.

033jK   Branch to K if (Xj) is Negative   NG Xj, K

29 – 21	20 – 18	17 – 0
033	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The branch decision for this instruction is based on the value of the sign bit in Xj.

Jump to K if: Bit 59 of Xj = 1 (negative)
Continue if: Bit 59 of Xj = 0 (positive)

This instruction branches on a negative result from either a fixed-point or a floating-point operation.

034jK Branch to K if (Xj) is in range IR Xj, K

29 — 21	20 — 18	17 — 0
034	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if:  (Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)
              (Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)
```

This instruction branches on a floating-point quantity within the floating-point range. The value of the coefficient is ignored in making this branch test. An underflow quantity is considered in range for purposes of this test.

035jK Branch to K if (Xj) is out of range OR Xj, K

29 — 21	20 — 18	17 — 0
035	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if:  (Xj) = 3777 xxxx xxxx xxxx xxxx (positive overflow)
               (Xj) = 4000 xxxx xxxx xxxx xxxx (negative overflow)
```

036jK Branch to K if (Xj) is definite DF Xj, K

29 — 21	20 — 18	17 — 0
036	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of Xj. The program sequence continues only on the following conditions. The branch to address K occurs for all other cases.

```
Continue if:  (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
              (Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)
```

This instruction branches on a floating-point quantity that may be out of range but is still defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

037jK Branch to K if (Xj) is indefinite ID Xj, K

29 — 21	20 — 18	17 — 0
037	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on the content of the Xj register. The branch to address K occurs only on the following conditions. The current program sequence continues for all other cases.

```
Jump to K if:  (Xj) = 1777 xxxx xxxx xxxx xxxx (positive indefinite)
               (Xj) = 6000 xxxx xxxx xxxx xxxx (negative indefinite)
```

This instruction branches on a floating-point quantity that is not defined. The value of the coefficient is ignored in making this branch test. An overflow quantity or an underflow quantity is considered defined for purposes of this test.

04ijK Branch to K if (Bi) = (Bj) EQ Bi, Bj, K

29 — 24	23 — 21	20 — 18	17 — 0
04	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. The branch to address K occurs only if the two quantities are identical on a bit-by-bit comparison basis. The current program sequence continues for all other cases.

This instruction branches on an index equality test. A quantity consisting of all 0's and a quantity consisting of all 1's are not equal for this test.

05ijK   Branch to K if (Bi) $\neq$ (Bj)   NE Bi, Bj, K

29 24	23 21	20 18	17 0
05	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of the Bi and Bj registers. The program sequence continues only if the two quantities are identical on a bit-by-bit comparison basis. The branch to address K occurs for all other cases.

This instruction branches on an index inequality test. A quantity consisting of all 0's and a quantity consisting of all 1's are not equal for this test.

06ijK   Branch to K if (Bi) $\geq$ (Bj)   GE Bi, Bj, K

29 24	23 21	20 18	17 0
06	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is greater than or equal to the content of Bj. The current program sequence continues if the content of Bi is less than Bj.

This instruction branches on an index threshold test. A +0 quantity is considered greater than a -0 quantity.

07ijK   Branch to K if (Bi) < (Bj)   LT Bi, Bj, K

29 24	23 21	20 18	17 0
07	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. Execution of this instruction causes the program sequence to terminate with a jump to address K in CM or to continue with the current program sequence, depending on a comparison of the contents of Bi and Bj. Both quantities are treated as signed integers. The branch to address K occurs if the content of Bi is less than the content of Bj. The current program sequence continues if the content of Bi is greater than or equal to the content of Bj.

This instruction branches on an index threshold test. A +0 quantity is considered greater than a -0 quantity.

CP Block Copy Instructions

The block copy instructions (table 4-3) transfer 60-bit words between fields in CM and UEM.

Table 4-3. CP Block Copy Instructions

Opcode	Format	Instruction	Mnemonic
011	jK	Block copy (Bj + K) words from UEM to CM	RE Bj+K
012	jK	Block copy (Bj + K) words from CM to UEM	WE Bj+K

Block Copy

011jK   Block copy (Bj + K) words from UEM to CM   RE Bj + K

59	51	47 30	29 0
011	i	K	INST. FOR HALF EXIT

This instruction copies a block of Bj plus K consecutive words from unified extended memory (UEM) to CM. The source UEM address is X0 plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of X0; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of X0; bits 30 through 59 are ignored.

The destination CM address is either A0 plus RAC, or X0 plus RAC, depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is A0 plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of X0. Bits 51 through 59 must be set to 0; results are undefined if these bits are not 0.

The operation leaves Bj, X0, and A0 unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a maximum of 131 071 60-bit words. If Bj plus K is 0, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of X0 plus RAE is a 1, 0's are transferred, and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 011jK instruction is interrupted, it is restarted from the beginning.

This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

In standard addressing mode, 24 bits of X0 are checked against 23 bits of FLE with bit 23 of FLE equal to 0. In expanded addressing mode, 30 bits of X0 are checked against 29 bits of FLE with bit 29 equal to 0. If the X0 bits are greater than or equal to FLE, an address-out-of-range condition is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in chapter 5.

012jK   Block copy (Bj + K) words from CM to UEM   WE Bj + K

59	51	47 30	29 0
012	j	K	INST. FOR HALF EXIT

This instruction copies a block of Bj plus K consecutive words from CM to UEM. The source CM address is either A0 plus RAC or X0 plus RAC, depending on the setting of the block copy flag in the CYBER 170 exchange package. When the block copy flag is clear, the CM address is A0 plus RAC. When the block copy flag is set, the CM address is calculated using bits 30 through 50 of X0. Bits 51 through 59 must be set to 0; results are undefined if these bits are not 0.

The destination UEM address is X0 plus RAE where the bits used depend on the setting of the expanded addressing select flag in the CYBER 170 exchange package. If the flag is clear (UEM is in standard addressing mode), the UEM address is calculated using bits 0 through 22 of X0; bits 24 through 59 are ignored. If the flag is set (UEM is in expanded addressing mode), the UEM address is calculated using bits 0 through 28 of X0; bits 30 through 59 are ignored.

The operation leaves Bj, X0, and A0 unchanged. Bj and K are both signed 18-bit one's complement numbers, making it possible to transfer a maximum of 131 071 60-bit words. If Bj plus K is 0, the instruction acts as a 60-bit pass instruction.

If bit 21 or 22 of the result of X0 plus RAE is a 1, 0's are transferred and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case, the next instruction is taken from parcel 0 of the next instruction word. If execution of the 012jK instruction is interrupted, it is restarted from the beginning.

This instruction is illegal if it does not start in parcel 0 or the UEM enable flag in the CYBER 170 exchange package is clear.

In standard addressing mode, 24 bits of X0 are checked against 23 bits of FLE with bit 23 of FLE equal to 0. In expanded addressing mode, 30 bits of X0 are checked against 29 bits of FLE with bit 29 equal to 0. If the X0 bits are greater than or equal to FLE, an address-out-of-range condition is detected.

If Bj plus K is negative, an address range error exit takes place. If the source field and the destination field overlap in physical memory, the final contents of the destination field are undefined.

For further information, refer to Block Copy Instructions in chapter 5.

CP Shift Instructions

The shift instructions (table 4-4) shift the Xi 60-bit word through the number of bit positions determined from a computed shift count.

Table 4-4. CP Shift Instructions

Opcode	Format	Instruction	Mnemonic
20	ijk	Left shift (Xi) by jk	LXi jk
22	ijk	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj Xk
21	ijk	Right shift (Xi) by jk	AXi jk
23	ijk	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj Xk

Left Shift

20ijk Left shift (Xi) by jk LXi jk

14 9	8 6	5 0
20	i	jk

This instruction reads one operand from Xi, shifts the 60-bit word left circularly by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end of the 60-bit word are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A sample computation is listed in octal notation to illustrate the operation performed.

Initial (Xi) = 2323 6600 0000 0000 0111

jk = 12 (octal)

Final (Xi) = 7540 0000 0000 0022 2464

This instruction, together with instruction 21, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, use instruction 22 or 23.

22ijk	Left shift (Xk) nominally (Bj) places to Xi	LXi Bj, Xk

14   9 8   6 5   3 2   0

22	i	j	k

This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is left-shifted circularly the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is right-shifted with sign extension the number of bit positions designated by the value in Bj. Bj bit 17 determines the sign of Bj.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced towards the lowest-order positions. The bits shifted off the lower end are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. An example of a positive shift count resulting in a left-circular shift is as follows:

(Xk) = 2323 6600 0000 0000 0111

(Bj) = 00 0012

(Xi) = 7540 0000 0000 0022 2464

An example of the right shift with sign extension is as follows:

(Xk) = 1327 6000 0000 3333 2422

(Bj) = 77 7771

(Xi) = 0013 2760 0000 0033 3324

If Bj bits 6 through 10 are different from Bj bit 17 and Bj bit 17 is set, the shift count is greater than 63 (decimal) places right, and a result of +0 is returned to Xi. Bj bits 11 through 16 are not tested by this instruction.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

Right Shift

21ijk Right shift (Xi) by jk AXi jk

14 9	8 6	5 0
21	i	jk

This instruction reads one operand from Xi, shifts the 60-bit word right with sign extension by jk bit positions, and writes the resulting 60-bit word back into the same Xi register. The j and k designators are treated as a single 6-bit positive integer operand in this instruction.

A right shift with sign extension implies that the bit pattern in the 60-bit word is displaced toward the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. An example of a positive operand is as follows:

Initial (Xi) = 2004 7655 0002 3400 0004

jk = 30 (octal)

Final (Xi) = 0000 0000 2004 7655 0002

An example of a negative operand is as follows:

Initial (Xi) = 6000 4420 2222 0000 5643

jk = 10 (octal)

Final (Xi) = 7774 0011 0404 4440 0013

This instruction, together with instruction 20, may be used whenever a data word is to be shifted by a predetermined amount. If the amount of shift is derived in the execution of the program, use instruction 22 or 23.

23ijk	Right shift (Xk) nominally (Bj) places to Xi	AXi Bj, Xk

14 9	8 6	5 3	2 0
23	i	j	k

This instruction reads a 60-bit operand from Xk, shifts the data either left or right as specified by the content of Bj, and writes the resulting 60-bit word into Xi. If the value in Bj is positive, the data is right-shifted with sign extension the number of bit positions designated by the value in Bj. If the value in Bj is negative, the data is left-shifted circularly the number of bit positions designated by the value in Bj. Bj bit 17 determines the sign of Bj.

A left-circular shift implies that the bit pattern in the 60-bit word is displaced towards the highest-order bit positions. The bits shifted off the upper end are inserted in the lowest-order bit positions in the same sequence. The resulting 60-bit word has the same quantity of bits with values of 1 and 0 as in the original operand.

A right shift with sign extension implies that the bit pattern in the 60-bit words is displaced towards the lowest-order bit positions. The bits shifted off the lower end of the word are discarded. The highest-order bit positions are filled with copies of the original sign bit.

Two sample computations are listed in octal notation to illustrate the operation performed. The following example contains a positive shift count resulting in a right shift with sign extension.

(Xk) = 1327 6000 0000 3333 2422

(Bj) = 00 0006

(Xi) = 0013 2760 0000 0033 3324

The following example contains a negative shift count resulting in a left-circular shift.

(Xk) = 2323 6600 0000 0000 0111

(Bj) = 77 7765

(Xi) = 7540 0000 0000 0022 2464

If Bj bits 6 through 10 are different from Bj bit 17, and Bj bit 17 is clear, the shift count is greater than 63 (decimal) places right, and a result of +0 is returned to Xi. This instruction does not test Bj bits 11 through 16.

This instruction is used when the amount of shift is derived in the computation. The instruction is also used for correcting the coefficient of a floating-point number when the exponent has been unpacked into a B register.

CP Logical Instructions

The logical instructions (table 4-5) perform logical (Boolean) operations in the X registers.

Table 4-5. CP Logical Instructions

Opcode	Format	Instruction	Mnemonic
12	ijk	Logical sum of (Xj) and (Xk) to Xi	BXi Xj+Xk
16	ijk	Logical sum of (Xj) with complement of (Xk) to Xi	BXi -Xk+Xj
13	ijk	Logical difference of (Xj) and (Xk) to Xi	BXi Xj-Xk
17	ijk	Logical difference of (Xj) with complement of (Xk) to Xi	BXi -Xk-Xj
11	ijk	Logical product of (Xj) and (Xk) to Xi	BXi Xj*Xk
15	ijk	Logical product of (Xj) with complement of (Xk) to Xi	BXi -Xj*Xj

Logical Sum

12ijk   Logical sum of (Xj) and (Xk) to Xi   BXi Xj + Xk

14 9	8 6	5 3	2 0
12	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 0123 7777 7777 4567 1110

This instruction merges portions of a 60-bit word into a composite word during data processing.

16ijk   Logical sum of (Xj) with complement of (Xk) to Xi   BXi -Xk + Xj

14 9	8 6	5 3	2 0
16	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical sum of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj) = 0000 7777 0123 4567 1010

(Xk) = 0123 4567 7777 0000 1100

(Xi) = 7654 7777 0123 7777 7677

This instruction merges portions of a 60-bit word into a composite word during data processing.

Logical Difference

13ijk Logical difference of (Xj) and (Xk) to Xi BXi Xj -Xk

14 9	8 6	5 3	2 0
13	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical difference of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 0000 3210 7654 7777 0110

This instruction compares bit patterns or complements bit patterns during data processing.

17ijk Logical difference of (Xj) with complement of (Xk) to Xi BXi -Xk - Xj

14 9	8 6	5 3	2 0
17	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical difference of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible combinations that may occur.

(Xj) = 0123 7777 0123 4567 1010

(Xk) = 0123 4567 7777 3210 1100

(Xi) = 7777 4567 0123 0000 7667

This instruction compares bit patterns or complements bit patterns during data processing.

Logical Product

11ijk   Logical product of (Xj) and (Xk) to Xi   BXi Xj * Xk

14 … 9	8 … 6	5 … 3	2 … 0
11	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the two operands. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0077 7700 1100

(Xi) = 0123 4000 0023 4500 1000

This instruction extracts portions of a 60-bit word during data processing.

15ijk   Logical product of (Xj) with complement of (Xk) to Xi   BXi -Xk * Xj

14 … 9	8 … 6	5 … 3	2 … 0
15	i	j	k

This instruction reads operands from two X registers, operates on them to form a result, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. The result delivered to Xi is the bit-by-bit logical product of the value in Xj and the complement of the value in Xk. Each of the 60 bits in Xj is compared with the corresponding bit in Xk to form a single bit in Xi. A sample computation is listed in octal notation to illustrate the operation performed and includes the four possible bit combinations that may occur.

(Xj) = 7777 7000 0123 4567 1010

(Xk) = 0123 4567 0007 7700 1100

(Xi) = 7654 3000 0120 0067 0010

This instruction extracts portions of a 60-bit word during data processing.

CP Floating-Point Arithmetic Instructions

The floating-point instructions (table 4-6) perform arithmetic operations on floating-point numbers.

Table 4-6. CP Floating-Point Instructions

Opcode	Format	Instruction	Mnemonic
30	ijk	Floating sum of (Xj) and (Xk) to Xi	FXi Xj+Xk
32	ijk	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj+Xk
34	ijk	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj+Xk
31	ijk	Floating difference of (Xj) and (Xk) to Xi	FXi Xj-Xk
33	ijk	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj-Xk
35	ijk	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj-Xk
40	ijk	Floating product of (Xj) and (Xk) to Xi	FXi Xj*Xk
41	ijk	Round floating product of (Xj) and (Xk) to Xi	RXi Xj*Xk
42	ijk	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj*Xk
44	ijk	Floating divide (Xj) by (Xk) to Xi	FXi Xj/Xk
45	ijk	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

Floating Sum

30ijk Floating sum of (Xj) and (Xk) to Xi FXi Xj + Xk

14 — 9	8 — 6	5 — 3	2 — 0
30	i	j	k

This instruction reads operands from two X registers, operates on them to form a floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The upper half of the result is then selected as a coefficient and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the sum is right-shifted one place, and the exponent is increased by one.

If the two operands have unlike signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form.

When the difference between the exponents is greater than 128 (decimal), the shifted sign bit is extended to the entire shifted operand. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

32ijk	Floating double-precision sum of (Xj) and (Xk) to Xi	DXi Xj + Xk

14–9	8–6	5–3	2–0
32	i	j	k

This instruction reads operands from two X registers, operates on them to form a double-precision, floating-point sum, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The sum of the quantities in Xj and Xk is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The two coefficients are then added to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted by one place, and the exponent is increased by 1. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

34ijk	Round floating sum of (Xj) and (Xk) to Xi	RXi Xj + Xk

14–9	8–6	5–3	2–0
34	i	j	k

This instruction reads operands from two X registers, operates on them to form a rounded floating-point sum, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point sum is a single-precision floating-point sum with a round bit (or bits) inserted before the add operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is inserted in the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least-significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have unlike signs. The second round bit is inserted before the coefficient is shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

Floating Difference

31ijk	Floating difference of (Xj) and (Xk) to Xi	FXi Xj - Xk

14 9	8 6	5 3	2 0
31	i	j	k

This instruction reads operands from two X registers, operates on them to form a floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The upper half of the result is then selected and packed along with the larger exponent to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted one place, and the exponent is increased by one.

If the two operands have like signs, the result coefficient may have leading zeros. No normalize operation is built into this instruction to correct this situation. A separate normalize instruction must be programmed if the result is to be kept in a normalized form. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

33ijk	Floating double-precision difference of (Xj) and (Xk) to Xi	DXi Xj - Xk

14 ... 9	8 ... 6	5 ... 3	2 ... 0
33	i	j	k

This instruction reads operands from two X registers, operates on them to form a double-precision, floating-point difference, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The two operands are unpacked from floating-point format, and the exponents are compared. The coefficient with the smaller exponent is right-shifted by the difference of the two exponents such that both coefficients are the same significance. The Xk coefficient is then subtracted from the Xj coefficient to form a 96-bit result. The lower half of the result is then selected and packed along with the larger exponent minus 48 (decimal) to form the result sent to Xi. If coefficient overflow occurs, the result is right-shifted one place, and the exponent is increased by one.

Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

35ijk	Round floating difference of (Xj) and (Xk) to Xi	RXi Xj - Xk

14 — 9	8 — 6	5 — 3	2 — 0
35	i	j	k

This instruction reads operands from two X registers, operates on them to form a rounded floating-point difference, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result of subtracting the quantity in Xk from the quantity in Xj is delivered to Xi in floating-point format and is not necessarily normalized.

The round floating-point difference is a single-precision, floating-point difference with a round bit (or bits) inserted before the subtract operation takes place. A round bit is always inserted in the coefficient with the larger exponent. If the two exponents are equal, the round bit is added to the coefficient for Xk. The round bit is equal to the complement of the sign bit and is inserted immediately to the right of the lowest-order bit in the coefficient. This has the effect of increasing the magnitude of the coefficient by one-half of the least-significant bit. A second round bit is inserted in a corresponding manner to the other coefficient if both operands are normalized or have like signs. The second round bit is inserted before the coefficient is shifted by the difference of the exponents. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

Floating Product

40ijk   Floating product of (Xj) and (Xk) to Xi   FXi Xj * Xk

14 9	8 6	5 3	2 0
40	i	j	k

This instruction reads operands from two X registers, operates on them to form a floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most-significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in floating-point calculations where rounding of operands is not desired, such as in multiple-precision arithmetic and in calculations involving error analysis. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

41ijk	Round floating product of (Xj) and (Xk) to Xi	RXi Xj * Xk

14 ... 9	8 ... 6	5 ... 3	2 ... 0
41	i	j	k

This instruction reads operands from two X registers, operates on them to form a rounded floating-point product, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The result is delivered to Xi in floating-point format. If both operands are normalized, the result is also normalized. If both operands are not normalized, the result is not normalized.

The two operands are unpacked from floating-point format. The exponents are added with a correction factor to determine the exponent for the result. The coefficients are multiplied as signed integers to form a 96-bit integer product. A rounding bit is added to bit position 46 of this product. The upper half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most-significant bit. The upper 48 bits are read from the double-precision product register. Leading zeros occur in this result coefficient.

This instruction is used in single-precision, floating-point calculations. For multiple-precision calculations, the 40 and 42 instructions must be used. Infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

42ijk	Floating double-precision product of (Xj) and (Xk) to Xi	DXi Xj * Xk

14 9	8 6	5 3	2 0
42	i	j	k

This instruction reads operands from two X registers, operates on them to form a double-precision, floating-point product, and delivers the lower half of this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format and are not necessarily normalized. The lower half of the double-precision product is delivered to Xi in floating-point format and is not necessarily normalized.

The operands are not rounded in this operation. The two operands are unpacked from floating-point format. The exponents are added to determine the exponent for the result. The result exponent is exactly 48 less than the exponent for a 40 instruction. The coefficients are multiplied as signed integers to form a 96-bit integer product. The lower half of this product is extracted to form the coefficient for the result. If the original operands are normalized and the double-precision product has only 95 significant bits, a 1-bit left shift is done to normalize the result coefficient. The resulting exponent is reduced by one count in this case.

If both operands are not normalized, the resulting double-precision product has less than 96 significant bits. No test is made for the position of the most-significant bit. The lower 48 bits are always read from the 96-bit product register.

This instruction is used in multiple-precision, floating-point calculations. This instruction also provides for integer multiplication capabilities where both operands have an exponent value of plus or minus zero, and neither coefficient has been normalized. The integer result sent to Xi is 48 bits with 60-bit sign extension. If the result exceeds 48 bits, the hardware does not detect an overflow. An overflow check can be made by executing a 40 instruction using the same two operands. If the result is nonzero, overflow is then indicated. An integer multiply operation is not intended for use with normalized operands. Infinite (3777xxx...x and 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands cause corresponding exit conditions to set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

Floating Divide

44ijk Floating divide (Xj) by (Xk) to Xi FXi Xj/Xk

14 9	8 6	5 3	2 0
44	i	j	k

This instruction reads operands from two X registers, operates on them to form a floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating-point format. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the exponent subtraction causes an underflow or overflow, an underflow or overflow result is returned even with the occurrence of a divide fault.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault causes an indefinite result to be returned to Xi.

This instruction is used in floating-point calculations where rounding of operands is not desired. In multiple-precision division, this instruction must be followed by a multiplication of the quotient by the divisor and subtracted from the dividend to reconstruct the remainder.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

45ijk	Round floating divide (Xj) by (Xk) to Xi	RXi Xj/Xk

14 9	8 6	5 3	2 0
45	i	j	k

This instruction reads operands from two X registers, operates on them to form a rounded floating-point quotient, and delivers this result to a third X register. The operands for this instruction are in Xj and Xk. These operands are in floating-point format. The result of dividing the content of Xj by the content of Xk is delivered to Xi. If both operands are normalized, the quotient is also normalized. The remainder from the division process is discarded.

The two operands are unpacked from floating-point format in this operation. The exponents are subtracted with a correction factor to determine the exponent for the result. The coefficient from Xj is positioned in a dividend register. The Xj quantity is modified by inserting a 2525...25 round pattern below the lowest-order bit of the dividend coefficient. The coefficient from Xk is trial-subtracted repeatedly from the dividend. The quotient bits are assembled in a quotient register. When 48 bits of the quotient are assembled, they are packed with the result exponent into floating-point format and delivered to Xi.

If the dividend is not normalized, the quotient cannot be normalized. However, the quotient is correct even though there may be leading zeros in the coefficient. If the divisor is not normalized, the quotient may be incorrect. If the coefficient for the content of Xj is larger than the coefficient for the content of Xk by a factor of two or more, a divide fault occurs. A divide fault causes an indefinite result to be returned to Xi.

This instruction is used in single-precision, floating-point calculations where rounding of operands is desired to reduce truncation errors.

If infinite (3777xxx...x or 4000xxx...x) or indefinite (1777xxx...x or 6000xxx...x) operands are used, corresponding exit conditions are set in the CP for exit mode action.

For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

CP Jump Instructions

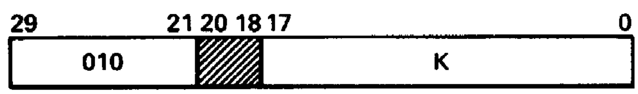
The jump instructions (table 4-7) allow departure from sequential instruction execution.

Table 4-7. CP Jump Instructions

Opcode	Format	Instruction	Mnemonic
010	xK	Return jump to K	RJ
02	ixK	Jump to (Bi) + K	JP

Jump

010xK   Return jump to K   RJ K



This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction writes a special word into CM at relative address K. The current program sequence then terminates by a jump to address K plus 1. The word stored in memory contains a jump instruction which causes an unconditional jump to the address of this return jump instruction plus 1.

This instruction calls a subroutine and inserts execution of the subroutine between execution of this instruction word and the following instruction word. Instructions appearing after the return jump instruction in the instruction word are not executed. The called subroutine exit must be at address K. The called subroutine entrance address must be K plus 1.

This instruction stores a 60-bit word at address K in memory. The upper half of this word contains an unconditional jump (0400) instruction with an address that is equal to the current program address plus 1. The lower half of the stored word is all 0's. The octal digits in the stored word then appear as illustrated with the x field indicating the location of the current program address plus 1.

K	0400x	xxxxx	00000	00000	Subroutine exit
K + 1	yyyyy	yyyyy	yyyyy	yyyyy	Subroutine entrance

02ixK   Jump to (Bi) + K   JP Bi + K

29–24	23–21	20–18	17–0
02	i		K

This two-parcel instruction uses the lower-order 18 bits as operand K. The instruction causes the current program sequence to terminate with a jump to address Bi plus K in CM.

This instruction allows computed branch point destinations. This is the only instruction in which a computed parameter can specify a program branch destination address. All other jump instructions have preassigned destination addresses.

The quantities in Bi and operand K are added in an 18-bit one's complement mode. The result is treated as an 18-bit positive integer that specifies the beginning address in CM for the new program sequence. The remaining instructions, if any, in the instruction word do not execute.

CP Exchange Jump Instructions

The exchange jump instructions (table 4-8) exchange the current process registers (formatted as an exchange package) with another set stored in CM, and do the following:

- When executed with CP in Virtual State monitor mode, the processor switches from monitor to job mode.
- When executed in Virtual State job mode, the processor switches from mob to monitor mode and the system call bit sets in the monitor condition register (MCR 10).

In either case, the P register stored in the outgoing exchange package points to the next instruction that would have executed if the exchange had not occurred.

This instruction can cause the following exception conditions.

- Environment specification error.
- System call.

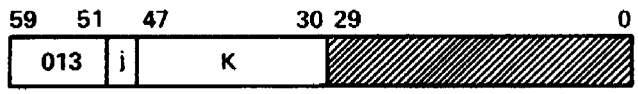
Refer to chapter 5 for programming information.

Table 4-8. CP Exchange Jump Instructions

Opcode	Format	Instruction	Mnemonic
013	jK	Central exchange jump to (Bj) +K (CYBER 170 monitor flag set)	XJ Bj+K
013	xx	Monitor exchange jump to MA (CYBER 170 monitor flag clear)	XJ

Exchange Jump

013jK	Central exchange jump to (Bj) +K when CYBER 170 MF set	XJ Bj + K
013xx	Monitor exchange jump to MA when CYBER 170 MF clear	XJ



This instruction must start at parcel 0. Also, a CYBER 170 exchange package must be ready at address Bj plus K or at address MA.

This instruction stores P plus 1 into the outgoing CYBER 170 exchange package in hardware and then exchanges this CYBER 170 exchange package with the CYBER 170 exchange package stored in memory. If the CYBER 170 MF is set at the beginning of the instruction, the incoming CYBER 170 exchange package starts at absolute address Bj plus K. If the CYBER 170 MF is clear at the beginning, then the j and K fields of the instruction are ignored, and the incoming CYBER 170 exchange package starts at absolute address MA, which is obtained from the outgoing CYBER 170 exchange package. In either case, the CYBER 170 MF is toggled, and the outgoing CYBER 170 exchange package is stored beginning at the same CM address from where the incoming CYBER 170 exchange package is obtained. Also, the jump is always to relative address P, parcel 0, from the new CYBER 170 exchange package. Refer to CYBER 170 Exchange Jump in chapter 5.

CP Compare/Move Instructions

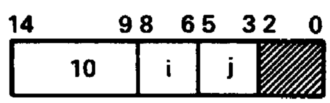
The compare/move instructions (table 4-9) move characters from one CM location to another and compare fields of characters either directly or through a collate table. The transmit instructions move words from one CM register to another.

Table 4-9. CP Compare/Move Instructions

Opcode	Format	Instruction	Mnemonic
10	ijx	Transmit (Xj) to Xi	BXi Xj
14	ixk	Transmit complement of (Xk) to Xi	BXi -Xk
464	jK	Move indirect	IM
465		Move direct	DM
466		Compare collated	CC
467		Compare uncollated	CU

Transmit

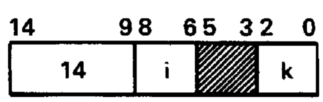
10ijx   Transmit (Xj) to Xi   BXi Xj



This instruction transfers a 60-bit word from Xj into Xi.

This instruction moves data from one X register to another X register. No logical function is performed on the data.

14ixk   Transmit complement of (Xk) to Xi   BXi -Xk



This instruction reads a 60-bit word from Xk, complements the word, and writes the result into Xi.

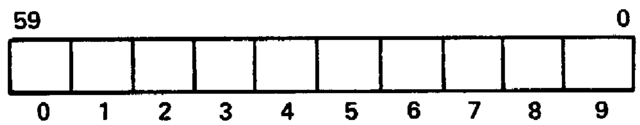
This instruction changes the sign of a fixed-point or floating-point quantity. The instruction also inverts an entire 60-bit field during data processing.

Compare/Move

The compare/move instructions (also referred to as CMU instructions) are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

CMU instructions must appear in parcel 0 or they are treated as illegal instructions.

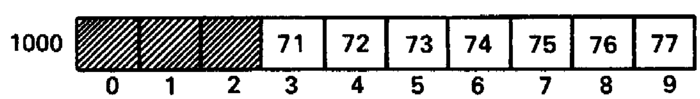
Data fields consisting of 6-bit characters may start or end with any character position (offset) of the ten 6-bit positions in each word. The character positions are designated as follows:



For move instructions, a K1 designator specifies which CM word contains the first character of the source data field, and a C1 designator specifies the character position (offset) of the first character. The K2 designator specifies the CM location in which the first character of the result data field is placed, and the C2 designator specifies the first character position. For compare instructions, both data field addresses specify source fields.

Example:

If the instruction is K1=1000 and C1=3, the first character of the source field is in position 3 of location 1000.



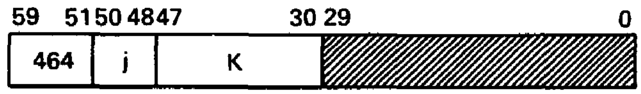
Therefore, the first character of the source field is 71.

An address is out of range if C1 or C2 is greater than 9, K1 plus N1 is greater than the program field length for CM (FLC), or K2 plus N2 is greater than FLC. N1 equals the number of CM references made to the source data field starting at K1, and N2 equals the number of CM references made to the result data field starting at K2. When an address-out-of-range condition occurs, the CMU instruction is not executed.

LL is the lower 4 bits, and LU is the upper 9 bits of the field length designator in numbers of characters. The maximum length of the data fields for the move direct and the compare instructions is 127 (177_8) characters. The maximum data field length for the move indirect instruction is 8191 (17777_8) characters. If L (LU and LL combined) is 0, the instruction becomes a pass.

For overlapping move instructions, the address of the source field (specified by K1) must be greater than the address of the result field (specified by K2) to provide proper field overlap. If K1 is less than K2, part of the source field is changed during execution. The amount of change is determined by the number of CM conflicts encountered. Overlapping fields should not contain more than 377 (octal) characters because an exchange jump interrupts any compare/move operation having a decremented field length greater than 377 (octal).

464jK Move indirect IM Bj + K

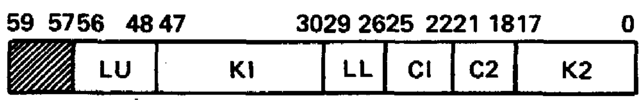


Any instructions located in the lower two parcels of the instruction word do not execute.

Bj plus K specifies a relative address in CM for the following descriptor word.

The descriptor word specifies the movement of the source field to the result field. The movement is from left to right through the field. Register X0 clears at the end of the execution.

465 Move direct DM



This instruction moves the source field to the result field as specified by the 60-bit instruction word. The field length is limited to a 7-bit count.

466 Compare collated CC

59 57	56 48	47 30	29 26	25 22	21 18	17 0
465	LU	K1	LL	C1	C2	K2

This instruction compares the field designated by K1,C1 with the field designated by K2,C2 as specified by the 60-bit instruction word.

The compare is from left to right through the fields until two unequal characters are found. These two characters are then collated and referenced in the collate table beginning at address A0 (table 4-10). If the table values found for the two unequal characters are equal, the compare continues until another pair of characters is unequal or until the field length is exhausted. If the table values found for the two unequal characters are unequal, X0 is set prior to instruction termination as follows:

- If field K1 is greater than field K2, set X0 to 0000 0000 0000 0000 0xxx.
- If field K1 is equal to field K2, set X0 to 0000 0000 0000 0000 0000.
- If field K1 is less than field K2, set X0 to 7777 7777 7777 7777 7yyy where yyy is the complement of xxx.

The value of the three octal numbers xxx that are stored in X0 is determined by the equation L minus N equals xxx (L is the length of the field, and N is the number of pairs of characters that were collated equal prior to instruction termination). In other words, xxx is the number of pairs of characters not yet compared plus 1.

The A0 register contains the starting word address of an 8-word, 64-character collate table (table 4-10). This table must have been previously stored in consecutive CM locations.

The collated value of a character is found by examining the collate table. The upper 3 bits of the character to be collated are added to A0 to obtain the relative address of the word containing the collated value. The lower 3 bits of the character to be collated specify the character address of the collated value.

Example:

Suppose the character under examination is an octal 63. The 6 is added to the A0 to form the word address. The 3 is used to pick the correct character from that word. The value of 63 is 63 in the collate table.

Table 4-10. Collate Table

Address	Collating Character Locations								
A0	00	01	02	03	04	05	06	07	xx	xx
A0+1	10	11	12	13	14	15	16	17	xx	xx
A0+2	20	21	22	23	24	25	26	27	xx	xx
A0+3	30	31	32	33	34	35	36	37	xx	xx
A0+4	40	41	42	43	44	45	46	47	xx	xx
A0+5	50	51	52	53	54	55	56	57	xx	xx
A0+6	60	61	62	63	64	65	66	67	xx	xx
A0+7	70	71	72	73	74	75	76	77	xx	xx

467 Compare uncollated CU

59 51	50 48	47 30	29 26	25 22	21 18	17 0
466	LU	K1	LL	C1	C2	K2

This instruction is similar to the 466 instruction except that the collate table is not used. The X0 register is set when the first pair of unequal characters is encountered or when the field length is exhausted.

CP Set Instructions

Table 4-11 lists the CP set instructions. Opcodes 50 through 57 obtain operands from CM for computation and deliver the results back into CM. The remaining opcodes operate on B or X registers only.

Table 4-11. CP Set Instructions

Opcode	Format	Instruction	Mnemonic
50	ijK	Set Ai to (Aj) + K	SAi Aj+K
51	ijK	Set Ai to (Bj) + K	SAi Bj+K
52	ijK	Set Ai to (Xj) + K	SAi Xj+K
53	ijK	Set Ai to (Xj) + (Bk)	SAi Xj+Bk
54	ijK	Set Ai to (Aj) + (Bk)	SAi Aj+Bk
55	ijK	Set Ai to (Aj) - (Bk)	SAi Aj-Bk
56	ijK	Set Ai to (Bj) + (Bk)	SAi Bj+Bk
57	ijK	Set Ai to (Bj) - (Bk)	SAi Bj-Bk
60	ijK	Set Bi to (Aj) + K	SBi Aj+K
61	ijK	Set Bi to (Bj) + K	SBi Bj+K
62	ijK	Set Bi to (Xj) + K	SBi Xj+K
63	ijK	Set Bi to (Xj) + (Bk)	SBi Xj+Bk
64	ijK	Set Bi to (Aj) + (Bk)	SBi Aj+Bk
65	ijK	Set Bi to (Aj) - (Bk)	SBi Aj-Bk
66	ijK	Set Bi to (Bj) + (Bk)	SBi Bj+Bk
67	ijK	Set Bi to (Bj) - (Bk)	SBi Bj-Bk
70	ijK	Set Xi to (Aj) + K	SXi Aj+K
71	ijK	Set Xi to (Bj) + K	SXi Bj+K
72	ijK	Set Xi to (Xj) + K	SXi Xj+K
73	ijK	Set Xi to (Xj) + (Bk)	SXi Xj+Bk
74	ijK	Set Xi to (Aj) + (Bk)	SXi Aj+Bk
75	ijK	Set Xi to (Aj) - (Bk)	SXi Aj-Bk
76	ijK	Set Xi to (Bj) + (Bk)	SXi Bj+Bk
77	ijK	Set Xi to (Bj) - (Bk)	SXi Bj-Bk
660	jK	Read CM at (Xk) to Xj	CRXj XK
670	jK	Write Xj into CM at (Xk)	CWXj Xk

Set Ai

50ijK   Set Ai to (Aj) + K   SAi Aj + K

29 24	23 21	20 18	17 0
50	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

51ijK   Set Ai to (Bj) + K   SAi Bj + K

29 24	23 21	20 18	17 0
51	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

52ijK   Set Ai to (Xj) + K   SAi Xj + K

29 24	23 21	20 18	17 0
52	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

53ijk   Set Ai to (Xj) + (Bk)   SAi Xj + Bk

14 9	8 6	5 3	2 0
53	i	j	k

This instruction reads operands from Xj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

54ijk Set Ai to (Aj) + (Bk) SAi Aj + Bk

14 … 9	8 … 6	5 … 3	2 … 0
54	i	j	k

This instruction reads operands from Aj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

55ijk Set Ai to (Aj) - (Bk) SAi Aj - Bk

14 … 9	8 … 6	5 … 3	2 … 0
55	i	j	k

This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

561jk   Set Ai to (Bj) + (Bk)   SAi Bj + Bk

14 — 9	8 — 6	5 — 3	2 — 0
56	i	j	k

This instruction reads operands from Bj and Bk, forms the sum of the operands, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the results back into CM.

57ijk   Set Ai to (Bj) - (Bk)   SAi Bj - Bk

14 — 9	8 — 6	5 — 3	2 — 0
57	i	j	k

This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Ai. If the i designator is nonzero, a reference is made to CM, using the result as the relative address. The type of reference is a function of the i designator value.

i = 0	No CM reference
i = 1,2,3,4,5	Read from CM to Xi
i = 6,7	Write into CM from Xi

This instruction obtains operands from CM for computation and delivers the result back into CM.

Set Bi

60ijK   Set Bi to (Aj) + K   SBi Aj + K

29 – 24	23 – 21	20 – 18	17 – 0
60	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. This instruction is for address modification in the increment registers.

61ijK   Set Bi to (Bj) + K   SBi Bj + K

29 – 24	23 – 21	20 – 18	17 – 0
61	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

62ijK   Set Bi to (Xj) + K   SBi Xj + K

29 – 24	23 – 21	20 – 18	17 – 0
62	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

63ijk Set Bi to (Xj) + (Bk) SBi Xj + Bk

14 9	8 6	5 3	2 0
63	i	j	k

This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

64ijk Set Bi to (Aj) + (Bk) SBi Aj + Bk

14 9	8 6	5 3	2 0
64	i	j	k

This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode.

65ijk Set Bi to (Aj) - (Bk) SBi Aj - Bk

14 9	8 6	5 3	2 0
65	i	j	k

This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is 0, this becomes a pass instruction.

66ijk   Set Bi to (Bj) + (Bk)   SBi Bj + Bk

14 9 8 6 5 3 2 0

66	i	j	k

This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Bi. The sum is formed in an 18-bit one's complement mode. If the i designator is 0, this becomes a read central memory instruction.

67ijk   Set Bi to (Bj) - (Bk)   SBi Bj - Bk

14 9 8 6 5 3 2 0

67	i	j	k

This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Bi. The difference is formed in an 18-bit one's complement mode. If the i designator is 0, this becomes a write central memory instruction.

Set Xi

70ijK Set Xi to (Aj) + K SXi Aj + K

29–24	23–21	20–18	17–0
70	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Aj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

71ijK Set Xi to (Bj) + K SXi Bj + K

29–24	23–21	20–18	17–0
71	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Bj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

72ijK Set Xi to (Xj) + K SXi Xj + K

29–24	23–21	20–18	17–0
72	i	j	K

This two-parcel instruction uses the lower-order 18 bits as operand K. This instruction reads an operand from Xj, forms the sum of the operand plus K, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

73ijk Set Xi to (Xj) + (Bk) SXi Xj + Bk

14 9	8 6	5 3	2 0
73	i	j	k

This instruction reads operands from Xj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

74ijk Set Xi to (Aj) + (Bk) SXi Aj + Bk

14 9	8 6	5 3	2 0
74	i	j	k

This instruction reads operands from Aj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

75ijk Set Xi to (Aj) - (Bk) SXi Aj - Bk

14 9	8 6	5 3	2 0
75	i	j	k

This instruction reads operands from Aj and Bk, subtracts the Bk operand from the Aj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

76ijk   Set Xi to (Bj) + (Bk)   SXi Bj + Bk

14 9	8 6	5 3	2 0
76	i	j	k

This instruction reads operands from Bj and Bk, adds the operands, and delivers the result to Xi. The sum is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

77ijk   Set Xi to (Bj) - (Bk)   SXi Bj - Bk

14 9	8 6	5 3	2 0
77	i	j	k

This instruction reads operands from Bj and Bk, subtracts the Bk operand from the Bj operand, and delivers the result to Xi. The difference is formed in an 18-bit one's complement mode. The 18-bit result is sign-extended by copying the highest-order bit of the result into the upper 42 bit positions in Xi.

Read/Write

660jk   Read central memory at (Xk) to Xj   CR Xj, Xk

```
14            65  32  0
|     660     | j | k |
```

This instruction loads into Xj the word at location (Xk), where Xk is a right-justified 21-bit relative word address. Bits 21 through 59 of Xk are ignored. If the 21 bits of Xk are greater than or equal to FLC, an address-out-of-range condition is detected.

670jk   Write Xj into central memory at (Xk)   CW Xj, Xk

```
14            65  32  0
|     670     | j | k |
```

This instruction stores Xj in location (Xk), where Xk is a 21-bit relative word address. Bits 21 through 59 of Xk are ignored. If the 21 bits of Xk are greater than or equal to FLC, an address-out-of-range condition is detected.

CP Normalize Instructions

The normalize instructions (table 4-12) perform normalizing operations in floating-point format and deliver the normalized result to Xi.

Table 4-12. CP Normalize Instructions

Opcode	Format	Instruction	Mnemonic
24	ijk	Normalize (Xk) to Xi and Bj	NXi Bj Xk
25	ijk	Round normalize (Xk) to Xi and Bj	ZXi Bj Xk

Normalize

24ijk Normalize (Xk) to Xi and Bj NXi Bj, Xk

14 9	8 6	5 3	2 0
24	i	j	k

This instruction reads one operand from Xk, performs a normalizing operation on this word in floating-point format, and delivers the normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The normalizing operation consists of repositioning the coefficient portion of the operand and then adjusting the exponent portion of the operand to leave the value of the result unaltered. The coefficient is shifted towards the higher-order bit positions of the word. The coefficient is shifted the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most-significant bit of the coefficient in the highest-order position. The exponent is then decreased by the number of bit positions shifted.

Two sample computations are listed in octal notation to illustrate the operation performed. The following example involves a positive floating-point number.

(Xk) = 2034 0047 6500 0000 2262

(Xi) = 2026 4765 0000 0022 6200

(Bj) = 00 0006

The following example involves a negative floating-point number.

(Xk) = 5743 7730 1277 7777 5515

(Xi) = 5751 3012 7777 7755 1577

(Bj) = 00 0006

Normalizing a number with either a +0 or a -0 coefficient sets a shift count in Bj to 48 (decimal) and enters Xi with +0. If Xk contains an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to 0. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action. If the exponent is less than negative 1777 with a zero coefficient, the contents of Xi and Bj are set to 0. For further information, refer to Floating-Point Arithmetic under CP Programming in chapter 5.

Round Normalize

25ijk   Round normalize (Xk) to Xi and Bj   ZXi Bj, Xk

14 ... 9	8 ... 6	5 ... 3	2 ... 0
25	i	j	k

This instruction reads one operand from Xk, performs a rounding and then a normalizing operation in floating-point format, and delivers the round normalized result to Xi. In addition, a positive integer shift count is sent to Bj. This shift count is the number of bit positions of shift required to normalize the original operand coefficient.

The rounding operation consists of adding a bit to the coefficient portion of the operand in a bit position immediately below the least-significant bit position. This round bit has a value equal to the complement of the operand sign bit. The result increases the magnitude of the coefficient by one-half the value of the least-significant bit.

The normalizing operation consists of repositioning the coefficient and adjusting the exponent to leave the value of the resulting floating-point quantity unaltered. The coefficient is shifted towards the higher-order bit positions. The round bit is shifted along with the coefficient. The displacement is the minimum number of bit positions required to make bit 47 different from sign bit 59. This places the most-significant bit of the coefficient in the highest-order bit position. The exponent is decreased by the number of bit positions shifted.

Two sample computations are listed in octal notation to illustrate the normalizing operation performed.

An example that involves a positive floating-point number is as follows.

(Xk) = 2034 0047 6500 0000 2262

(Xi) = 2026 4765 0000 0022 6420

(Bj) = 00 0006

The following example involves a negative number.

(Xk) = 5743 7730 1277 7777 5515

(Xi) = 5751 3012 7777 7755 1537

(Bj) = 00 0006

If Xk contains either an infinite quantity (3777xxx...x or 4000xxx...x) or an indefinite quantity (1777xxx...x or 6000xxx...x), no shift takes place. The content of Xk is copied to Xi, and Bj is set to 0. Corresponding infinite and indefinite exit conditions are also set in the CP for exit mode action.

Refer to Floating-Point Arithmetic under CP Programming in chapter 5.

CP Pass Instructions

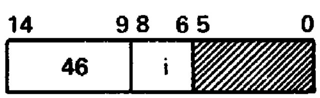
The pass instructions (table 4-13) perform no operation and are used for filling words to get the next instruction properly positioned.

Table 4-13. CP Pass Instructions

Opcode	Format	Instruction	Mnemonic
460	xx	Pass	NO
461	xx	Pass	
462	xx	Pass	
463	xx	Pass	

Pass

460xx Pass NO
thru 463xx



These instructions fill program instruction words where necessary to match jump destinations with word boundaries. The j and k designators are ignored, and a nonzero value has no effect in this instruction.

CP Illegal Instructions

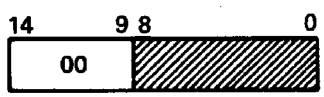
The illegal instructions (table 4-14) cause an exchange to CYBER 170 monitor mode, when in CYBER 170 job mode, and cause a jump to executive state when in CYBER 170 monitor mode.

Table 4-14. CP Illegal Instructions

Opcode	Format	Instruction	Mnemonic
00xxx		Error exit to MA or interrupt to executive mode	
017	jK	Illegal instruction (Trap 180)	RT
014	jK	Read one word from UEM to Xj	RXj Xk
015	jK	Write one word from Xj to UEM	WXj Xk

Error Exit

00xxx Error exit to MA when CYBER 17 MF clear
Interrupt to executive mode when CYBER 170 MF set PS



This instruction causes an illegal instruction error exit. CYBER 170 MF is the hardware monitor flag. Refer to Illegal Instructions in chapter 5.

Illegal Instruction

017jk Illegal Instruction

Refer to Illegal Instructions in chapter 5.

Illegal Read/Write

014jk   Read one word from (Xk + RAE) to Xj   RXj Xk

14 6	5 3	2 0
014	j	k

This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction reads the 60-bit word from UEM location Xk plus RAE into Xj. Xk is less than FLE.

The number of bits checked for an address-out-of-range condition varies, depending on the addressing mode of UEM. In standard addressing mode, 24 bits of Xk are checked against 23 bits of FLE with bit 23 of FLE equal to 0. In expanded addressing mode, 30 bits of Xk are checked against 29 bits of FLE with bit 29 of FLE equal to 0. If Xk is greater than or equal to FLE, an address-out-of-range condition is detected.

015jk   Write one word from Xj to (Xk + RAE)   WXj Xk

14 6	5 3	2 0
015	j	k

This instruction is illegal if the UEM enable flag in the CYBER 170 exchange package is clear. This instruction writes the 60-bit word from Xj into the UEM location Xk plus RAE. Xk is less than FLE.

The number of bits checked for an address-out-of-range condition varies, depending on the addressing mode of UEM. In standard addressing mode, 24 bits of Xk are checked against 23 bits of FLE with bit 23 of FLE equal to 0. In expanded addressing mode, 30 bits of Xk are checked against 29 bits of FLE with bit 29 of FLE equal to 0. If Xk is greater than or equal to FLE, an address-out-of-range condition is detected.

CP Mask Instruction

Form Mask

43ijk Form mask of jk bits to Xi MXi jk

14 9	8 6	5 3	2 0
43	i	j	k

This instruction generates a masking word using the j and k designators as parameters. No operands are read from operating registers. The j and k designators are treated as a single, 6-bit octal quantity to designate the width of the masking field. A field of 1's, beginning at the highest-order end of the word, is extended downward on a background of 0's. The completed masking word consists of 1 bits in the highest-order jk bit positions and 0 bits in the remainder of the word. This masking word is then delivered to Xi. The following are sample parameters.

j = 2

k = 4

Xi = 7777 7760 0000 0000 0000

This instruction generates variable width masks for logical operations. This instruction, together with a shift instruction, generally creates an arbitrary field mask faster than reading a pregenerated mask from CM.

CP Pop Count Instruction

Population Count

47ixk   Population count of (Xk) to Xi   CXi Xk

14 … 9	8 … 6	5 … 3	2 … 0
47	i		k

This instruction reads one operand from Xk, counts the number of one bits in the operand, and stores the count in Xi. The count delivered to Xi is a positive integer. If the operand is all 1's, a count of 60 (decimal) is delivered to Xi. If operand is all zeros, a 0's word is delivered to Xi.

CP Read Free Running Counter Instruction

Read Free-Running Counter

016jk   Read free running counter   RC Xj

14 … 6	5 … 3	2 … 0
016	j	k

This instruction transfers the current contents of the 48-bit free running counter to the Xj register. The leftmost 12 bits of Xj are set to 0. The k field is ignored.

This instruction is a single parcel instruction that can be located in any parcel.

PP Instruction Descriptions

The peripheral processor (PP) instruction set comprises the following eight subgroups.

- Load/Store.
- Arithmetic.
- Logical.
- Replace.
- Branch.
- Central Memory Access.
- Input/Output.
- Other.

PP Instruction Formats

Figure 4-2 shows PP instruction formats. PP instructions are 16 or 32 bits long. In instruction descriptions, the operation code is given either by two or three octal digits. The third digit, when used, indicates the state of the s-bit (0 or 1) in I/O instructions (refer to table 4-15).

The upper 4 bits of the PP instructions must be 0 to ensure that the instructions operate as defined in this chapter.

Table 4-15. PP Nomenclature

Term	Description
Opcode	Specifies instruction operation code.
s	Specifies I/O instruction subcode.
c	Specifies channel number.
A	Refers to the A register (arithmetic register) or the content of the A register.
(A)	Refers to the content of the word at the CM address specified by the A register.
P	Refers to the P register or to the content of the P register (program address register).
R	Refers to the R register or to the content of the R register (relocation register).
(d)	Refers to the content of the word at the PP memory address specified by the d field (direct mode).
((d))	Refers to the content of the word at the PP memory address specified by the content of the word at the PP memory address specified by the d field (indirect mode).
m + (d)	Refers to the PP memory address specified by the m field indexed by the content of the word at the PP memory addressed specified by the d field.
(m + (d))	Refers to the content of the word at the PP memory address specified by the m field indexed by the content of the word at the PP memory address specified by the d field (memory mode).

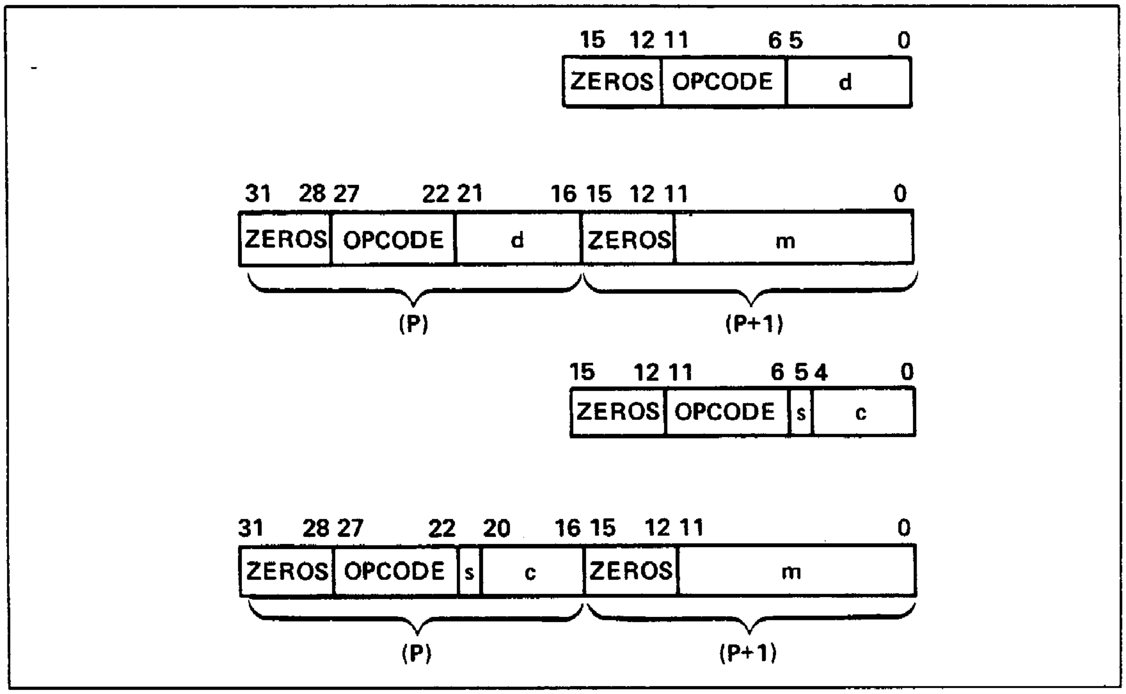


Figure 4-2. PP Instruction Formats

PP Data Format

Figure 4-3 shows PP data format and how 12-bit data is packed into 64-bit CM words or unpacked from 64-bit CM words.

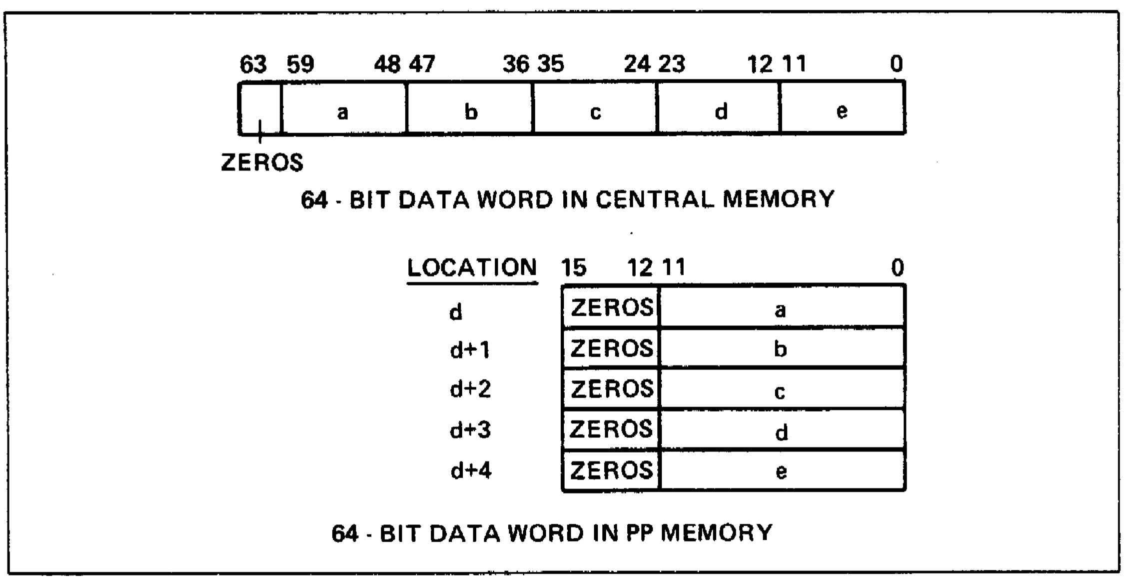


Figure 4-3. PP Data Format

PP Relocation Register Format

Figure 4-4 shows PP relocation (R) register format. This register is loaded-from/stored-into PP memory by instructions 24 and 25 (load/store R register).

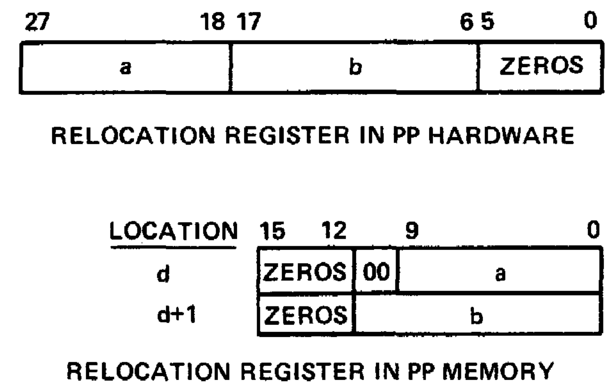


Figure 4-4. PP Relocation (R) Register Format

PP Load/Store Instructions

Load and store instructions (table 4-16) transfer 6-, 10-, 12-, and 18-bit quantities between the PP A register and the PP memory.

Table 4-16. PP Load/Store Instructions

Opcode	Format	Instruction	Mnemonic
14	d	Load d	LDN d
15	d	Load complement d	LCH d
20	dm	Load dm	LDC m,d
24	d	Load R	LRD d
30	d	Load (d)	LDD d
40	d	Load ((d))	LDI d
50	dm	Load (m+(d))	LDM m,d
25	d	Store R	SRD d
34	d	Store (d)	STD d
44	d	Store ((d))	STI d
54	dm	Store (m+(d))	STM m,d

Load

14d Load d LDN d

15 12	11 6	5 0
00	14	d

This instruction clears the A register and loads d. The upper 12 bits of A are 0.

15d Load complement d LCN d

15 12	11 6	5 0
00	15	d

This instruction clears the A-register and loads the complement of d. The upper 12 bits of A are 1.

20dm Load dm LDC dm

31 28	27 22	21 16	15 12	11 0
00	20	d	00	m
	(P)			(P+1)

This instruction clears the A register and loads an 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

24d Load R register LRD d

15 12	11 6	5 0
00	24	d

Figure 4-4 shows R register format. If d is not equal to 0, this instruction loads the upper 10 bits of the R register (bits 18-27) from the rightmost 10 bits of PP memory location d. The 12 bits contained in PP memory location d plus 1 are loaded into the next 12 bits of the R register (bits 6 through 17). If d equals 0, the instruction is a pass.

30d   Load (d)   LDD d

15 12	11 6	5 0
00	30	d

This instruction clears the A register and loads the content at location d. The upper 6 bits of A are 0.

40d   Load ((d))   LDI d

15 12	11 6	5 0
00	40	d

This instruction clears the A register and loads a 12-bit quantity that is obtained by indirect addressing. The upper 6 bits of A are 0. Location d is read from PPM, and the word read is used as the operand address.

50dm   Load (m + (d))   LDM m,d

31 28	27 22	21 16	15 12	11 0
00	50	d	00	m
(P)			(P+1)	

This instruction clears the A register and loads a 12-bit quantity. The upper 6 bits of A are 0's. The 12-bit operand is obtained by indexed direct addressing.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

Store

25d   Store R register   SRD   d

15 12	11 6	5 0
00	25	d

Figure 4-4 shows R register format. If d is not equal to 0, this instruction stores the upper 10 bits of the R register (bits 18 through 27) into the rightmost 10 bits of PP memory location d. The 12 bits contained in PP memory location d plus 1 are stored into the next 12 bits of the R register (bits 6 through 17). If d equals 0, the instruction is a pass.

34d   Store (d)   STD   d

15 12	11 6	5 0
00	34	d

This instruction stores the lower 12 bits of the A register at location d.

44d   Store ((d))   STI   d

15 12	11 6	5 0
00	44	d

This instruction stores the lower 12 bits of the A register at the location specified by the content of location d.

54dm   Store (m + (d))   STM   m,d

31 28	27 22	21 16	15 12	11 0
00	54	d	00	m
(P)			(P+1)	

This instruction stores the lower 12 bits of the A register in the location determined by indexed direct addressing.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

PP Arithmetic Instructions

The PP arithmetic instructions (table 4-17) perform integer arithmetic using the PP A register contents as one operand, with the other operand specified by the instruction. The result replaces the original contents of A. The PP considers the operands as one's complement integers and performs the arithmetic in one's complement.

Table 4-17. PP Arithmetic Instructions

Opcode	Format	Instruction	Mnemonic
16	d	Add d	ADN d
21	dm	Add dm	ADC m,d
31	d	Add (d)	ADD d
41	d	Add ((d))	ADI d
51	dm	Add (m+(d))	ADM m,d
17	d	Subtract d	SBN d
32	d	Subtract (d)	SBD d
42	d	Subtract ((d))	SBI d
52	dm	Subtract (m+(d))	SBM m,d

Arithmetic Add

16d   Add d   ADN d

15 12	11 6	5 0
00	16	d

This instruction adds d (treated as a 6-bit positive quantity) to the content of the A register.

21dm   Add dm   ADC dm

31 28	27 22	21 16	15 12	11 0
00	21	d	00	m
(P)			(P+1)	

This instruction adds to the A register the 18-bit quantity consisting of d as the upper 6 bits and m as the lower 12 bits. The content of the location (P plus 1) which follows the present program address (P) is read to provide m.

31d Add (d) ADD d

15 12	11 6	5 0
00	31	d

This instruction adds the content at location d (treated as a 12-bit positive quantity) to the A register.

41d Add ((d)) ADI d

15 12	11 6	5 0
00	41	d

This instruction adds to the content of the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

51dm Add (m + (d)) ADM m,d

31 28	27 22	21 16	15 12	11 0
00	51	d	00	m
(P)			(P+1)	

This instruction adds the 12-bit operand (treated as a positive quantity) read by indexed direct addressing to the A register.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

Arithmetic Subtract

17d Subtract d SBN d

15 12	11 6	5 0
00	17	d

This instruction subtracts d (treated as a 6-bit positive quantity) from the content of the A register.

32d Subtract (d) SBD d

15 12	11 6	5 0
00	32	d

This instruction subtracts the content at location d (treated as a 12-bit positive quantity) from the A register.

42d Subtract ((d)) SBI d

15 12	11 6	5 0
00	42	d

This instruction subtracts from the A register a 12-bit operand (treated as a positive quantity) obtained by indirect addressing. Location d is read from PPM, and the word read is used as the operand address.

52dm Subtract (m + (d)) SBM m,d

31 28	27 22	21 16	15 12	11 0
00	52	d	00	m
	(P)		(P+1)	

This instruction subtracts the 12-bit operand (treated as a positive quantity) read by indexed direct addressing from the A register.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

PP Logical Instructions

The logical instructions (table 4-18) perform operations with one operand as the PP A register contents, and the other as specified by the instruction. The result replaces the original contents of A.

Table 4-18. PP Logical Instructions

Opcode	Format	Instruction	Mnemonic
10	d	Shift d	SHN d
13	d	Selective clear d	SCN d
11	d	Logical difference d	LMN d
23	dm	Logical difference dm	LMC m,d
33	d	Logical difference (d)	LMD d
43	d	Logical difference ((d))	LMI d
53	dm	Logical difference (m+(d))	LMM m,d
12	d	Logical product d	LPN d
22	dm	Logical product dm	LPC m,d

Shift

10d Shift d SHN d

15 12	11 6	5 0
00	10	d

This instruction shifts the content of the A register right or left d places. If d is positive (00 through 37), the shift is left circular. If d is negative (40 through 77), the shift is right circular (end-off with no sign extension). Thus, d equal to 06 requires a left-shift of six places; d equal to 71 requires a right-shift of six places.

Selective Clear

13d Selective clear d SCN d

15 12	11 6	5 0
00	13	d

This instruction clears any of the lower 6 bits of the A register where corresponding bits of d are 1. The upper 12 bits of A are not altered.

Logical Difference

11d Logical difference d LMN d

15 12	11 6	5 0
00	11	d

This instruction forms the bit-by-bit logical difference of d and the lower 6 bits of A in the register in A. This is equivalent to complementing individual bits of A that correspond to bits of d that are 1. The upper 12 bits of A are not altered.

23dm Logical difference dm LMC dm

31 28	27 22	21 16	15 12	11 0
00	23	d	00	m
(P)			(P+1)	

This instruction forms the bit-by-bit logical difference of the content of the A register and the 18-bit quantity dm in A. This is equivalent to complementing individual bits of A which correspond to bits of dm that are 1. The upper 6 bits of the quantity consist of d, and the lower 12 bits are the content of the location (P plus 1), which follows the present program address (P).

33d Logical difference (d) LMD d

15 12	11 6	5 0
00	33	d

This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the content at location d. This is equivalent to complementing individual bits of A that correspond to bits in location d that are 1's. The upper 6 bits are not altered.

43d   Logical difference ((d))   LMI d

15 12	11 6	5 0
00	43	d

This instruction forms in the A register the bit-by-bit logical difference of the lower 12 bits of the A register and the 12-bit operand read by indirect addressing. Location d is read from PPM, and the word read is used as the operand address. The upper 6 bits of A are not altered.

53dm   Logical difference (m + (d))   LMM m,d

31 28	27 22	21 16	15 12	11 0
00	53	d	00	m
(P)			(P+1)	

This instruction forms the bit-by-bit logical difference of the lower 12 bits of the A register and a 12-bit operand obtained by indexed direct addressing in the A register. The upper 6 bits of A are not altered.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

Logical Product

12d Logical product d LPN d

15 12	11 6	5 0
00	12	d

This instruction forms the bit-by-bit logical product of d and the lower 6 bits of the A register and leaves this quantity in the lower 6 bits of A. The upper 12 bits of A are 0.

22dm Logical product dm LPC dm

31 28	27 22	21 16	15 12	11 0
00	22	d	00	m
(P)			(P+1)	

This instruction forms the bit-by-bit logical product of the content of the A register and the 18-bit quantity dm in A. The upper 6 bits of this quantity consist of d, and the lower 12 bits are the content of the location (P plus 1), which follows the present program address (P).

PP Replace Instructions

The replace instructions (table 4-19) perform integer arithmetic with one operand as the contents of A and the other as specified by the instruction. The result replaces the original contents of A and the contents of the other operands location. The result stored in location d is either the rightmost 12 bits (for the normal instructions) or the rightmost 16 bits (for the long instructions) of the A register. Therefore, since A contains 18 bits, the value remaining in A cannot equal the value stored in PP memory location d. The PP considers the operands as one's complement integers and performs one's complement arithmetic.

Table 4-19. PP Replace Instructions

Opcode	Format	Instruction	Mnemonic
35	d	Replace add (d)	RAD d
36	d	Replace add 1 (d)	AOD d
45	d	Replace add ((d))	RAI d
46	d	Replace add 1 ((d))	AOI d
55	dm	Replace add (m+(d))	RAM m,d
56	dm	Replace add 1 (m+(d))	AOM m,d
57	dm	Replace subtract 1 (m+(d))	SOM m,d
37	d	Replace subtract 1 (d)	SOD d
47	d	Replace subtract 1 ((d))	SOI d

Replace Add

35d Replace add (d) RAD d

15 12 11 6 5 0

00	35	d

This instruction adds the quantity at location d to the content of the A register and stores the lower 12 bits of the result at location d. The result remains in A at the end of the operation, and the original content of A is destroyed.

36d Replace add 1 (d) AOD d

15 12 11 6 5 0

00	36	d

This instruction replaces the quantity at location d with its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

45d Replace add ((d)) RAI d

15 12 11 6 5 0

00	45	d

This instruction adds the operand, which is obtained from the location specified by the content at location d, to the content of the A register. The lower 12 bits of the sum replace the original operand. The result remains in A at the end of the operation.

46d Replace add 1 ((d)) AOI d

15 12	11 6	5 0
00	46	d

This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

55dm Replace add (m + (d)) RAM m,d

31 28	27 22	21 16	15 12	11 0
00	55	d	00	m
(P)			(P+1)	

This instruction adds the operand, which is obtained from the location determined by indexed direct addressing, to the A register. The lower 12 bits of the sum replace the original operand in PPM. The result remains in A at the end of the operation, and the original content of A is destroyed.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

56dm Replace add 1 (m + (d)) AOM m,d

31 28	27 22	21 16	15 12	11 0
00	56	d	00	m
(P)			(P+1)	

This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing, by its original value plus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

Replace Subtract

37d Replace subtract 1 (d) SOD d

15 12	11 6	5 0
00	37	d

This instruction replaces the quantity at location d with its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

47d Replace subtract 1 ((d)) SOI d

15 12	11 6	5 0
00	47	d

This instruction replaces the operand, which is obtained from the location specified by the content at location d, by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

57dm Replace subtract 1 (m + (d)) SOM m,d

31 28	27 22	21 16	15 12	11 0
00	57	d	00	m
(P)			(P+1)	

This instruction replaces the operand, which is obtained from the location determined by indexed direct addressing, by its original value minus 1. The result remains in the A register at the end of the operation, and the original content of A is destroyed.

In indexed direct addressing, the quantity m, which is read from PPM location P plus 1, serves as the base operand address to which the content of d is added. If d equals 0, the operand address is m, but if d is not equal to 0, m plus the content in d is the operand address. Therefore, location d may be used as an index quantity to modify operand addresses.

PP Branch Instructions

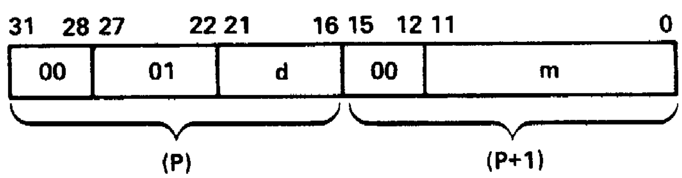
The branch instructions (table 4-20) allow departure from sequential instruction execution.

Table 4-20. PP Branch Instructions

Opcode	Format	Instruction	Mnemonic
01	dm	Long jump to m + (d)	LJM m,d
02	dm	Return jump to m + (d)	RJM m,d
03	d	Unconditional jump d	UJN d
04	d	Zero jump d	ZJN d
05	d	Nonzero jump d	NJN d
06	d	Plus jump d	PJN d
07	d	Minus jump d	MJN d
640	cm	Jump to m if channel c active	AJM m,c
650	cm	Jump to m if channel c inactive	IJM m,c
660	cm	Jump to m if channel c full	FJM m,c
661	cm	Jump to m if channel c error flag set	SFM m,40B+c
670	cm	Jump to m if channel c empty	EJM m,c
671	cm	Jump to m if channel c error flag clear	CFM m,40B+c

Long Jump

01dm Long jump to m + (d) LJM m,d



This instruction jumps to the address given by m plus the content of location d. If d equals 0, m is not modified.

Return Jump

02dm   Return jump to m + (d)   RJM m,d

31–28	27–22	21–16	15–12	11–0
00	02	d	00	m

This instruction jumps to the address given by m plus the content of location d. If d equals zero, m is not modified. The current program address (P) plus 2 is stored at the jump address. The next instruction starts at the jump address plus 1. The subprogram exits with a long jump or normal sequencing to the jump address minus 1, which in turn contains a long jump, 0100. This returns the original program address plus 2 to the P register.

Unconditional Jump

03d   Unconditional jump d   UJN d

15–12	11–6	5–0
00	03	d

This instruction provides an unconditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. The value of d is added to the current program address. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs. A deadstart is required to restart the PP.

Zero/Nonzero Jump

04d Zero jump d ZJN d

15 12	11 6	5 0
00	04	d

This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is 0, the jump is taken. If the content of A is nonzero, the next instruction executes from P plus 1. A -0 (777777) is treated as nonzero. For interpretation of d, refer to the 03 instruction.

05d Nonzero jump d NJN d

15 12	11 6	5 0
00	05	d

This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is nonzero, the jump is taken. If the content of A is 0, the next instruction executes from P plus 1. A -0 (777777) is treated as nonzero. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs. A deadstart is required to restart the PP.

Plus/Minus Jump

06d Plus jump d PJN d

15 12	11 6	5 0
00	06	d

This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the sign of the A register is positive, the jump is taken. If the sign of A is negative, the next instruction executes from P plus 1. A +0 is treated as a positive quantity. A -0 is treated as a negative quantity. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs. A deadstart is required to restart the PP.

07d Minus jump d MJN d

15 12	11 6	5 0
00	07	d

This instruction provides a conditional jump to any address up to 31 (decimal) locations forward or backward from the current program address. If the content of the A register is negative, the jump is taken. If the content of A is positive, the next instruction executes from P plus 1. A +0 is treated as a positive quantity. A -0 is treated as a negative quantity. If d is positive (01 through 37), 0001 through 0037 is added, and the jump is forward. If d is negative (40 through 76), 7740 through 7776 is added, and the jump is backward. When d equals 00 or 77, the PP hangs. A deadstart is required to restart the PP.

Jump To m

640cm   Jump to m if channel c active   AJM m,c

31 28	27 22		20 16	15 12	11 0
00	64	0	c	00	m
(P)				(P+1)	

If channel c is active, this instruction causes a jump to m; otherwise, it is a pass.

650cm   Jump to m if channel c inactive   IJM m,c

31 28	27 22		20 16	15 12	11 0
00	65	0	c	00	m
(P)				(P+1)	

This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is inactive. The next instruction is at P plus 2 if the channel is active.

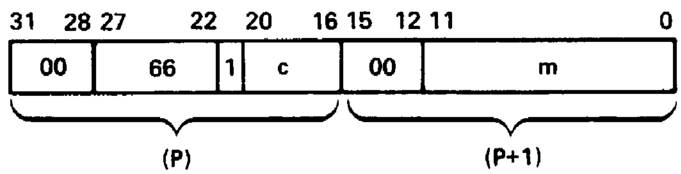
660cm   Jump to m if channel c full   FJM m,c

31 28	27 22		20 16	15 12	11 0
00	66	0	c	00	m
(P)				(P+1)	

This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel designated by c is full. The next instruction is at P plus 2 if the channel is empty.

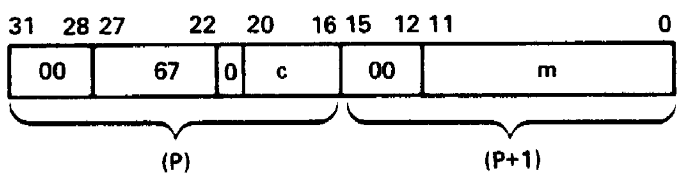
An input channel is full when the input equipment places a word in the channel and no PP has accepted that word. The channel is empty when a word has been accepted. An output channel is full when a PP places a word on the channel. The channel is empty when the output equipment accepts the word.

661cm Jump to m if channel c error flag set SFM m,c



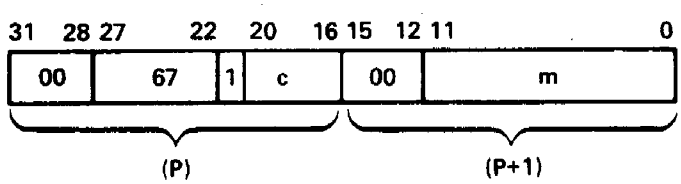
If the channel c error flag is set, this instruction clears the error flag and causes a jump to m. If this error flag is clear, the instruction is a pass. When m is set to P plus 2, the channel error flag is unconditionally cleared when the program reaches P plus 2.

670cm Jump to m if channel c empty EJM m,c



This instruction provides a conditional jump to a new address specified by m. The jump is taken if the channel specified by c is empty. The next instruction is at P plus 2 if the channel is full. An input channel is full when the input equipment places a word in the channel and no PP has accepted that word. The channel is empty when a word has been accepted. An output channel is full when a PP places a word on the channel. The channel is empty when the output equipment accepts the word.

671cm Jump to m if channel c error flag clear CFM m,c



If the channel c error flag is clear, this instruction causes a jump to m. If this error flag is set, the instruction clears the error flag and proceeds with the next instruction. When m is set to P plus 2, the channel error flag is unconditionally cleared when the program reaches P plus 2.

PP Central Memory Access Instructions

The PP central memory access instructions (table 4-21) provide the capability to read and write CM words to and from PP memory. The PPs have read access to all CM storage locations, while the OS bounds register controls write and exchange accesses. The IOU performs CM addressing with real memory word addresses. To address all locations in the larger CM sizes available, the IOU uses address relocation to modify the CM address in the A register of the PP. If bit 46 in A is 1 during a PP central memory read or write instruction, the IOU adds the R register contents to A register bits 47 through 63 to produce the CM address. If bit 46 of A is 0, the IOU does not perform address relocation but uses the A address. The R register contains an absolute 64-word starting boundary within CM. When relocation is desired, an absolute CM address is formed by concatenating six 0's to the rightmost end of the R contents and adding bits 47 through 63 of A.

Table 4-21. PP Central Memory Access Instructions

Opcode	Format	Instruction	Mnemonic
60	d	Central read from (A) to d	CRD d
61	dm	Central read (d) words from (A) to m	CRM m,d
62	d	Central write to (A) from d	CWD d
63	dm	Central write (d) words to (A) from m	CWM m,d

Central Read

60d Central read from (A) to d CRD d

15 12	11 6	5 0
00	60	d

This instruction disassembles one 60-bit word from central memory into five 12-bit words and stores these in five consecutive PP memory locations, beginning with the leftmost 12 bits of the 60-bit word.

The parameters of the transfer are as follows: If bit 17 of A is 0, A bits 0 through 16 contain the absolute address of the 60-bit word transferred. If bit 17 of A is 1, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in chapter 2, and PP Relocation Register Format at the beginning of this section on PP Instruction Descriptions. Field d gives the PP location that receives the first 12-bit word transferred. PP memory addressing is cyclic, and location 0000 follows location 7777.

61dm   Central read (d) words from (A) to m   CRM d,m

31 28	27 22	21 16	15 12	11 0
00	61	d	00	m
(P)			(P+1)	

PP location 0000 is used by hardware. This instruction disassembles 60-bit words from central memory into 12-bit words, and places these in consecutive PP memory locations, beginning with the leftmost 12 bits of the first 60-bit word.

The parameters of the transfer are as follows: If bit 17 of A is 0, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is 1, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in chapter 2, and PP Relocation Register Format under PP Instruction Descriptions. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location into which the first 12-bit word is placed.

This instruction stores P plus 1 into PP location 0000 before beginning the transfer. After the transfer is completed, the next instruction is taken from 1 plus whatever address is stored in location 0000. If the transfer overwrites location 0000, execution resumes at the location specified by (0000) plus 1 and results are undefined. (PP memory addressing is cyclic, and location 0000 follows location 7777.)

The A register is incremented by 1 after each 60-bit word is read from central memory. If the incrementing changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. Refer to Central Memory Addressing by PPs in chapter 5.

After the transfer is completed, the A register contains either the address of the last word transferred plus 1 (direct addressing) or the same address less the contents of the relocation address register (relocation addressing), except as follows: If the last word transferred is from a relative address 377776_8 and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus 1.

Central Write

62d Central write to (A) from d CWD d

15 12	11 6	5 0
00	62	d

This instruction assembles five 12-bit words from consecutive PP memory locations into one 60-bit word and stores the 60-bit word in central memory. The first 12-bit word is stored in the leftmost 12 bits of the 60-bit word. (PP memory addressing is cyclic, and location 0000 follows location 7777.)

The parameters of the transfer are as follows: If bit 17 of A is 0, A bits 0 through 16 contain the absolute address of the 60-bit word stored. If bit 17 of A is 1, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the 60-bit word stored. For further information, refer to R Register under Input/Output Unit in chapter 2, and PP Relocation Register Format under PP Instruction Descriptions. Field d gives the PP location of the first 12-bit word transferred. The transfer is subject to the CM bounds test.

63dm   Central write (d) words to (A) from m   CWM m,d

31–28	27–22	21–16	15–12	11–0
00	63	d	00	m
(P)			(P+1)	

Hardware uses PP location 0000. This instruction assembles 12-bit words from consecutive PP memory locations into 60-bit words and stores these in central memory. The first 12-bit word is stored in the leftmost 12 bits of the 60-bit word. (PP memory addressing is cyclic, and location 0000 follows location 7777.)

The parameters of the transfer are as follows: If bit 17 of A is 0, A bits 0 through 16 contain the absolute address of the first 60-bit word transferred. If bit 17 of A is 1, hardware adds relocation register R to zero-extended A bits 0 through 16 to obtain the absolute address of the first 60-bit word transferred. For further information, refer to R Register under Input/Output Unit in chapter 2 and in PP Relocation Register Format at the beginning of this section on PP Instruction Descriptions. PP location d must contain the number of 60-bit words transferred. Field m gives the PP location from where the first 12-bit word is obtained. The transfer is subject to the CM bounds test. This instruction stores P plus 1 into PP location 0000 before beginning the transfer. After the transfer is completed, the next instruction is taken from 1 plus whatever address is stored in location 0000.

The A register is incremented by 1 after each 60-bit word is written into central memory. If the incrementing changes A bit 17, the central memory addressing is switched between direct address and relocation address modes. Refer to Central Memory Addressing by PPs in chapter 5.

After the transfer is completed, the A register contains either the address of the last word transferred plus 1 (direct addressing) or the same address less the contents of the relocation address register (relocation addressing), except as follows: If the last word transferred is from a relative address 377776_8 and relocation is in effect, then the A register is cleared, and the value returned in A may not point to the last word transferred plus 1.

PP Input/Output Instructions

The PP input/output instructions (table 4-22) direct activity on the I/O channels. They select an external device and transfer data to or from that device. The instructions also determine whether a channel or external device is available and ready to transfer data. The preparatory steps ensure that the channels carry out an orderly data transfer. Each external device has a set of external function codes that the PP uses to establish operation modes, and to start and stop data transfer. The devices can also detect certain errors that are indicated to the controlling PP.

Table 4-22. PP Input/Output Instructions

Opcode	Format	Instruction	Mnemonic
641	cm	Test and set channel c flag	SCF m,40B+c
651	cm	Clear channel c flag	CCF c
70	d	Input to A from channel d	IAN d
71	dm	Input A words to m from channel d	IAM m,d
72	d	Output from A on channel d	OAN d
73	dm	Output (A) words from m on channel d	OAM m,d
74	d	Activate channel d	ACN d
75	d	Deactivate channel d	DCN d
76	d	Function A on channel d	FAN d
77	dm	Function m on channel d	FNC m,d

Test/Clear

641cm Test and set channel c flag SCF m,c

31 28 27 22 20 16 15 12 11 0

00	64	1	c	00	m

(P) (P+1)

If the channel c flag is set, this instruction causes a jump to m. If the channel c flag is clear, it sets this flag and continues with the next instruction. When m is set to P plus 2, the channel flag is unconditionally set when the program reaches P plus 2.

If two or more PPs simultaneously issue this instruction for the same channel, the conflict is resolved as follows:

If one of the competing channels is channel 17 (maintenance channel), the PP in the lowest physical level sees the true condition of the flag; the other conflicting PPs see the flag set (and hence take a jump). If the competing channel is any other channel, software must resolve the conflict. Any five consecutively numbered PPs (in the same barrel) issue instructions at different times.

651cm Clear channel c flag CCF m,c

31 28 27 22 20 16 15 12 11 0

00	65	1	c	00	m

(P) (P+1)

This instruction clears the channel c flag. The m field is required but is not used.

Input/Output

70d Input to A from channel d IAN d

15 12	11 6	5 0
00	70	d

This instruction transfers a word from input channel d to the lower 12 bits of the A register. The upper 6 bits of A are cleared to 0.

NOTE

If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive or is deactivated before a full is received, the instruction exits. The word is not accepted, and the A register clears.

71dm Input A words to m from channel d IAM m,d

31 28	27 22	21 16	15 12	11 0
00	71	d	00	m
(P)			(P+1)	

This instruction transfers a block of 12-bit words from input channel d to PPM. The first word goes to the PPM address specified by m. The A register holds the block length. A reduces by 1 as each word is read. The input operation completes when A equals 0 or the data channel becomes inactive. If the operation terminates by the channel becoming inactive, the next storage location in PPM is set to 0. However, the word count is not affected by this empty word. Therefore, A holds the block length minus the number of real data words read.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by 1 to hold the address for the next word as each word is stored.

NOTE

If this instruction executes when the data channel is inactive, no input operation is accomplished, and the program continues at P plus 2. However, the location specified by m is set to 0.

72d Output from A on channel d OAN d

15 12	11 6	5 0
00	72	d

This instruction transfers a word from the A register (lower 12 bits) to output channel d.

NOTE

If bit 5 of d is clear and the channel is inactive, this instruction hangs the PP, waiting for the channel to go active and full, if executed. If bit 5 of d is set and the channel is inactive, the program continues at P plus 1. The word is not transferred.

73dm Output A words from m on channel d OAM m,d

31 28	27 22	21 16	15 12	11 0
00	73	d	00	m
(P)			(P+1)	

This instruction transfers a block of words from PPM to channel d. The first word is read from the address specified by m. The A register holds the number of words to be sent. A reduces by 1 as each word is read. The output operation completes when A equals 0 or the channel becomes inactive.

During this instruction, address 0000 temporarily holds P while m is held in the P register. P advances by 1 to give the address of the next word as each word is read from the PPM.

NOTE

If this instruction executes when the data channel is inactive, no output operation is accomplished, and the program continues at P plus 2.

Activate/Deactivate

74d   Activate channel d   ACN d

15 12	11 6	5 0
00	74	d

This instruction activates the channel specified by d and sends the active signal on the channel to equipment connected to the channel. Activating a channel, which must precede a 70 through 73 instruction, prepares I/O equipment for the exchange of data.

NOTE

If this instruction executes when the data channel is already active and if bit 5 of d is set, the program continues at P plus 1. Otherwise, activating an already active channel causes the PP to wait until the channel goes inactive. The PP hangs if the channel does not go inactive.

75d Deactivate channel d DCN d

15 12	11 6	5 0
00	75	d

This instruction deactivates the channel specified by d. As a result, the I/O data transfer stops.

NOTES

If this instruction executes when the data channel is already inactive and bit 5 of d is set, the program continues at P plus 1. The channel remains inactive, and no inactive signal is sent to the I/O equipment. Deactivating an already inactive channel causes the PP to hang until the channel becomes active.

If an output instruction is followed by a disconnect instruction without first establishing that the input device (check for channel empty) has accepted the information, the last word transmitted may be lost.

Do not deactivate a channel before putting a useful program in the associated PP. PPs other than 0 are hung on an input instruction (71) after deadstart. Deactivating a channel after deadstart causes an exit to the address specified by the content of location 0000 plus 1 and execution of that program. If the channel is deactivated without a valid program in that PP, the PP executes whatever program was left in PPM. Therefore, the PP could run wild.

Function

76d Function A on channel d FAN d

15 12	11 6	5 0
00	76	d

This instruction sends the external function code in the lower 12 bits of the A register on channel d.

NOTE

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 1. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

77dm Function m on channel d FNC m,d

31 28	27 22	21 16	15 12	11 0
00	77	d	00	m
	(P)		(P+1)	

This instruction sends the external function code specified by m on channel d.

NOTE

If this instruction executes with bit 5 of d clear and the channel active, PP execution stops until a deadstart or another PP causes the channel to become inactive. If bit 5 of d is set and the channel is active, the program continues at P plus 2. Neither the function signal nor the function word transmits. The channel remains active, and execution continues.

Other IOU Instructions

Table 4-23 lists the other IOU instructions.

Table 4-23. Other IOU Instructions

Opcode	Format	Instruction	Mnemonic
00	xx	Pass	-
27	d	Pass	
260	x	Exchange Jump	EXN
261	x	Monitor exchange jump	MXN
262	x	Monitor exchange jump to MA	MAN

Pass

00xx Pass PSN

15 12	11 6	5 0
00	00	d

This instruction specifies that no operation is to be performed. The instruction provides a means of padding out a program.

27d Pass KPT d

15 12	11 6	5 0
00	27	d

This instruction is not an operation. However, it generates a pulse to a testpoint (keypoint) for optional monitoring by external equipment.

Exchange Jump

2600 Exchange jump EXN

15 12	11 6	5 0
00	26	00

This instruction causes an unconditional exchange jump in the CP, leaving the CP CYBER 170 monitor flag unaltered. The new CYBER 170 exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange is completed before proceeding with the next instruction.

2610 Monitor exchange jump MXN

15 12	11 6	5 0
00	26	10

If the CP is in the CYBER 170 monitor mode, this instruction is a pass. If the CP is in the CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to the CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at central memory location R plus A when the leftmost bit in A is set. When this bit is clear, A specifies the address. The PP waits until the exchange is completed before proceeding with the next instruction.

2620 Monitor exchange jump to MA MAN

15 12	11 6	5 0
00	26	20

If the CP is in CYBER 170 monitor mode, this instruction is a pass. If the CP is in CYBER 170 job mode, it causes a CYBER 170 exchange jump in the CP, switching the CP to CYBER 170 monitor mode (MF equals 1). The new CYBER 170 exchange package begins at the absolute address given in the MA field of the outgoing CYBER 170 exchange package. The PP waits until the exchange is completed before proceeding with the next instruction.

Instruction Execution Timing

Table 4-24 lists approximate execution times for the PP instructions. These times are listed with the assumption that no conflicts occur. The numbers in the timing notes column refer to the notes at the end of the table. Execution times are given in 250-ns major cycles.

NOTE

These execution times are approximations only and are subject to change without notice. Accurate timings can come only from benchmark tests. Control Data Corporation is not responsible for assumptions made based on the times listed here.

Table 4-24. PP Instruction Timing

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
00xx	Pass	1	-
01dm	Long jump to m + (d)	3	-
02dm	Return jump to m + (d)	4	-
03d	Unconditional jump d	1	-
04d	Zero jump d	1	-
05d	Nonzero jump d	1	-
06d	Plus jump d	1	-
07d	Minus jump d	1	-
10d	Shift d	1	-
11d	Logical difference d	1	-
12d	Logical product d	1	-
13d	Selective clear d	1	-
14d	Load d	1	-
15d	Load complement d	1	-
16d	Add d	1	-
17d	Subtract d	1	-
20dm	Load dm	2	-
21dm	Add dm	2	-
22dm	Logical product dm	2	-
23dm	Logical difference dm	2	-
24d	Load R register from (d) and (d) + 1	3	-
25d	Store R register at (d) and (d) + 1	4	-

(Continued)

Table 4-24. PP Instruction Timing (Continued)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
260x	Exchange jump	2	1
261x	Monitor exchange jump	2	1
262x	Monitor exchange jump to MA	2	1
27d	Pass	1	-
30d	Load (d)	2	-
31d	Add (d)	2	-
32d	Subtract (d)	2	-
33d	Logical difference (d)	2	-
34d	Store (d)	2	-
35d	Replace add (d)	4	-
36d	Replace add one (d)	5	-
37d	Replace subtract one (d)	5	-
40d	Load ((d))	3	-
41d	Add ((d))	3	-
42d	Subtract ((d))	3	-
43d	Logical difference ((d))	3	-
44d	Store ((d))	3	-
45d	Replace add ((d))	5	-
46d	Replace add one ((d))	6	-

Timing Notes:

1. No assembly-disassembly unit (ADU) conflicts and no outstanding CYBER 170 exchange jump request in the ADU.

(Continued)

Table 4-24. PP Instruction Timing (Continued)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
47d	Replace subtract one ((d))	6	-
50dm	Load (m + (d))	4	-
51dm	Add (m + (d))	4	-
52dm	Subtract (m + (d))	4	-
53dm	Logical difference (m + (d))	4	-
54dm	Store (m + (d))	4	-
55dm	Replace add (m + d))	6	-
56dm	Replace add one (m + (d))	7	-
57dm	Replace subtract one (m + (d))	7	-
60d	Central read from (A) to d	12	2
61dm	Central read (d) words from (A) to m	-	2,3
62d	Central write to (A) from d	6	2
63dm	Central write (d) words to (A) from m	-	2,4
640cm	Jump to m if channel c active	2	-
641cm	Test and set channel c flag	2	-
650cm	Jump to m if channel c inactive	2	-
651cm	Clear channel c flag	2	-
660cm	Jump to m if channel c full	2	-

Timing Notes:

2. No ADU conflicts. No central memory conflicts. Add a possible trip due to resynchronization (CM read instructions only).

3. Seven major cycles for instruction set-up and instruction exit. Five major cycles for every CM word.

4. Six major cycles for instruction set-up and instruction exit. Five major cycles for every CM word.

(Continued)

Table 4-24. PP Instruction Timing (Continued)

Instruction Code	Description	Execution Time in 250-ns Cycles	Timing Notes
661cm	Jump to m if channel c error flag set	2	-
670cm	Jump to m if channel c empty	2	-
671cm	Jump to m if channel c error flag clear	2	-
70d	Input to A from channel d	2	-
71dm	Input A words to m from channel d	-	5
72d	Output from A on channel d	2	-
73dm	Output (A) words from m on channel d	-	5
74d	Activate channel d	2	-
75d	Deactivate channel d	2	-
76d	Function A on channel d	2	-
77dm	Function m on channel d	2	-

Timing Notes:

5. Five major cycles for instruction set-up and exit. One major cycle per word (nonconflict case) or two major cycles per word (conflict case).

   Nonconflict case occurs when two PPs communicating to each other are not in the slot at the same time.

   Conflict case occurs when two PPs communicating with each other are in the slot at the same time.

5

Programming Information

Programming Information 5

This chapter contains special programming information about the CP, CM, PPs, system console, real-time clock, two-port multiplexer, and maintenance channel.

CP Programming

CYBER 170 Exchange Jump

The CP operates in either CYBER 170 job mode, which is interruptable, or CYBER 170 monitor mode, which is not interruptable. A hardware flag called the CYBER 170 monitor flag (MF) indicates the mode in which the CP is executing a job.

The CP uses a CYBER 170 exchange jump operation to switch from CYBER 170 job mode to CYBER 170 monitor mode and back again. The execution of a CYBER 170 exchange jump permits the CP to send pertinent information from the operating and control registers to CM and permits CM to send new information to the same registers. The information that flows from and into the operating and control registers during a CYBER 170 exchange jump is called a CYBER 170 exchange package (figure 5-1).

The CP 013 instruction and the PP 2600, 2610, and 2620 instructions initiate a CYBER 170 exchange jump operation. A CYBER 170 exchange jump instruction starts or interrupts the CP and provides CM with the first address of a 16-word exchange package. For the 013 instruction with MF set (CP in monitor mode), the starting address of the CYBER 170 exchange package is Bj plus K. With MF clear (CP in job mode), the address is the monitor address (MA). For the 2600 instruction, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A. For the 2610 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is A plus R when bit 17 of the A register is set. When this bit is clear, the address is A. For the 2620 instruction with MF set, the instruction is a pass. With MF clear, the CYBER 170 exchange package address is MA of the outgoing CYBER 170 exchange package.

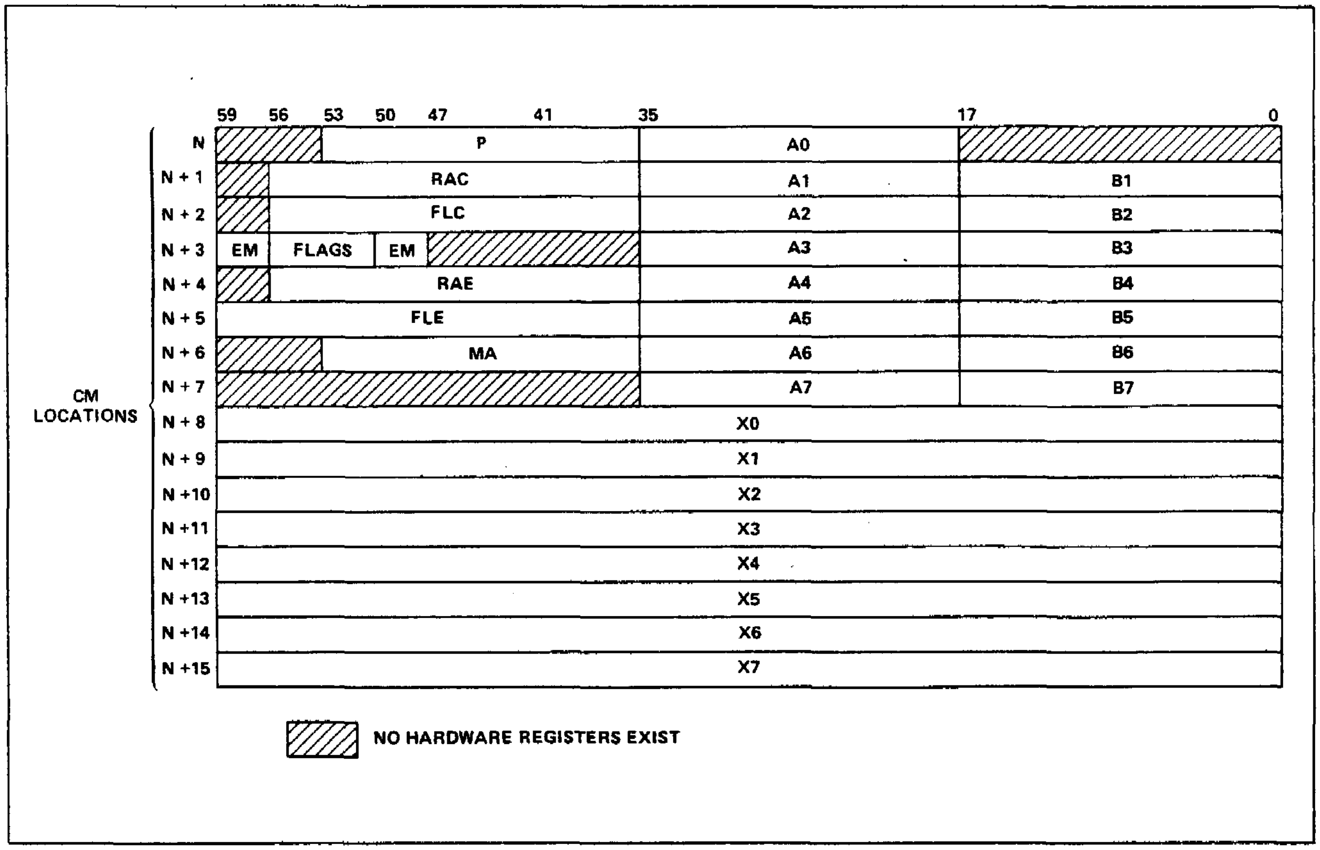


Figure 5-1. CYBER 170 Exchange Package

The CYBER 170 exchange package contains the following registers which provide information for program execution.

- 18-bit program address (P) register.
- 21-bit reference address for CM (RAC) register.
- 21-bit field length for CM (FLC) register.
- 6-bit exit mode (EM) register.
- 6-bit flag register.
- 21- or 24-bit reference address for UEM (RAE); 21 bits with lower 6 bits assumed to be 0 in standard addressing mode; 24 bits right-shifted with 6 bits assumed to be 0's in expanded addressing mode.
- 21- or 24-bit field length for UEM (FLE); 21 bits in standard addressing mode and 24 bits in expanded addressing mode; lower 6 bits are assumed to be 0.
- 18-bit monitor address (MA) register.
- Initial contents of eight 60-bit X registers.
- Initial contents of eight 18-bit A registers.
- Initial contents of 18-bit B registers B1 through B7; B0 contains a constant 0.

The time that a particular CYBER 170 exchange package resides in the CP hardware registers is the execution interval. The execution interval begins with a CYBER 170 exchange jump that swaps the CYBER 170 exchange package information in CM with the information contained in the CP registers. The execution interval ends with the next CYBER 170 exchange jump.

Executive State

The executive state uses a combination of hardware, software, and microcode to handle the following items.

- System initialization.
- Compare/move instructions.
- Software errors and unimplemented instructions that occur in CYBER 170 monitor mode.
- Processor-detected hardware errors.
- Hardware integrity verification (diagnostics).

In general, executive state determines the cause of an interrupt and decides whether to return the CP to the interrupted mode, to halt the CP, or to simulate a CYBER 170 exchange and return control to CYBER 170 monitor mode. Refer to Error Response in this chapter.

Floating-Point Arithmetic

Format

Floating-point arithmetic expresses a number in the form kBn.

k = Coefficient

B = Base number

n = Exponent or power to which the base number is raised

B is assumed to be 2 for binary-coded quantities. In the 60-bit, floating-point format (figure 5-2), the binary point is considered to be to the right of the coefficient. The lower 48 bits express the integer coefficient, which is the equivalent of 15 decimal digits. The sign of the coefficient is separated from the rest of the coefficient and appears in the highest-order bit of the packed word. Negative numbers are represented in one's complement notation. The exponent is biased by complementing the exponent sign bit.

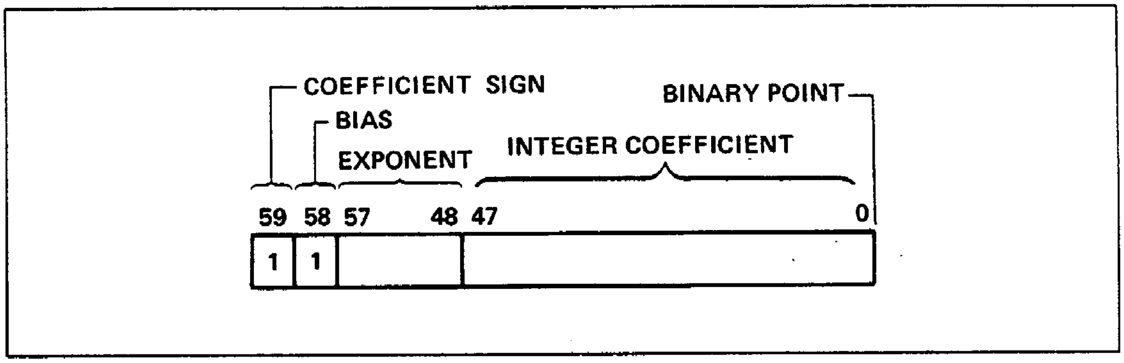


Figure 5-2. Floating-Point Format

Table 5-1 summarizes the configurations of bits 58 and 59 and the implications regarding signs of the possible combinations.

Table 5-1. Bits 58 and 59 Configurations

Bit 59	Bit 58	Coefficient Sign	Exponent Sign
0	1	Positive	Positive
0	0	Positive	Negative
1	0	Negative	Positive
1	1	Negative	Negative

Packing

Packing refers to the conversion of numbers in the form kB^n to floating-point format. A shortcut method of packing exponents can be derived by considering the representation of -0 and +0 exponents. Assuming a positive coefficient, 0 exponents are packed as follows:

+0 exponent: 2000x,...,x

-0 exponent: 1777x,...,x

Since positive exponents are expressed in true form, begin with a bias of 2000 (+0) and add the magnitude of the exponent. The range of positive exponents is 0000 through 1777. In packed form, the range is 2000 through 3777.

When the coefficient is negative, the packed positive exponent is complemented to become 5777 through 4000.

Negative exponents are expressed in complement form by beginning with a bias of 1777 (-0) and then subtracting the magnitude of the exponent. The range of negative exponents is negative 0000 through negative 1777. In packed form, the range is 1777 through 0000.

When the coefficient is negative, the packed negative exponent is complemented to become 6000 through 7777.

Examples of packed and unpacked floating-point numbers are shown in octal notation to illustrate the packing process. Examples 1 and 2 are different forms of the integer positive 1. Example 3 is positive 100 (decimal), and example 4 is negative 100 (decimal). Examples 5 and 6 are large and small positive numbers. The unpacked values are shown as they might appear in the X and B registers prior to a pack operation.

The packed -0 exponent is not used for normal operation. Instead, 1777 is used to indicate the special error condition of indefinite.

1.	Unpacked coefficient	0000	0000	0000	0000	0001
	Unpacked exponent	00	0000			
	Packed format	2000	0000	0000	0000	0001
2.	Unpacked coefficient	0000	4000	0000	0000	0000
	Unpacked exponent	77	7720			
	Packed format	1720	4000	0000	0000	0000
3.	Unpacked coefficient	0000	6200	0000	0000	0000
	Unpacked exponent	77	7726			
	Packed format	1726	6200	0000	0000	0000
4.	Unpacked coefficient	7777	1577	7777	7777	7777
	Unpacked exponent	77	7726			
	Packed format	6051	1577	7777	7777	7777
5.	Unpacked coefficient	0000	4771	3000	0044	7021
	Unpacked exponent	00	1363			
	Packed format	3363	4771	3000	0044	7021
6.	Unpacked coefficient	0000	6301	0277	4315	6033
	Unpacked exponent	77	6210			
	Packed format	0210	6301	0277	4315	6033

Overflow

Overflow of the floating-point range is indicated by an exponent value of positive 1777 (3777 or 4000 in packed form). This is the largest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial overflow. However, further computation using this result generates an overflow.

A complete overflow occurs whenever a result requires an exponent larger than positive 1777. In this case, a complete overflow value results. This result has a positive 1777 exponent and a zero coefficient. The sign of the coefficient is the same as that which generates if the result had not overflowed the floating-point range.

Underflow

Underflow of the floating-point range is indicated by an exponent value of negative 1777 (0000 or 7777 in packed form). This is the smallest exponent value that can be represented in the floating-point format. This exponent value may result from the calculation in which this exponent value, together with the computed coefficient value, is a correct representation of the result. This situation is called a partial underflow. Further computation using this result may be detected as an underflow.

A complete underflow occurs whenever a result requires an exponent smaller than negative 1777. In this case, a complete underflow value results. This result has a negative 1777 exponent and a zero coefficient. The complete underflow indicator is a word of all 0's, and it is the same as a zero word in integer format.

Indefinite

An indefinite result indicator generates whenever the calculation is unresolvable. An example is division when the divisor is 0 and the dividend is also 0. Another example is multiplication of an overflow number times an underflow number. The indefinite result indicator is a value that cannot occur in normal floating-point calculations. This indicator corresponds to a -0 exponent and a 0 coefficient (177770,...,0 in packed form).

Any indefinite indicator used as an operand generates an indefinite result no matter what the other operand value is. Although indefinite indicators always generate with a positive sign, they may occur as operands with a negative sign.

Nonstandard Operands

In summary, the special operand forms in octal are:

Positive overflow (+ ∞)	3777x,...,x
Negative overflow (- ∞)	4000x,...,x
Positive indefinite (+IND)	1777x,...,x
Negative indefinite (-IND)	6000x,...,x
Positive underflow (+0)	0000x,...,x
Negative underflow (-0)	7777x,...,x

Tables 5-2 through 5-5 indicate the resulting forms when various combinations of underflow, overflow, and indefinite forms are used in floating-point operations. The designations W and N are defined as follows:

W Any word except + ∞ and + IND

N Any word except + ∞ , + IND, and + 0

Table 5-2. Xj Plus Xk (30, 32, 34 Instructions)

		Xk			
		W	+ ∞	− ∞	+IND
Xj	W		+ ∞	− ∞	IND
	+ ∞	+ ∞	+ ∞	IND	IND
	− ∞	− ∞	IND	− ∞	IND
	±IND	IND	IND	IND	IND

Table 5-3. Xj Minus Xk (31, 33, 35 Instructions)

		Xk			
		W	+ ∞	− ∞	±IND
	W		− ∞	+ ∞	IND
	+ ∞	+ ∞	IND	+ ∞	IND
Xj	− ∞	− ∞	− ∞	IND	IND
	±IND	IND	IND	IND	IND

Table 5-4. Xj Multiplied by Xk (40, 41, 42 Instructions)

		Xk						
		+N	−N	+0	−0	+ ∞	− ∞	±IND
	+N			0	0	+ ∞	− ∞	IND
	−N			0	0	− ∞	+ ∞	IND
	+0	0	0	Integer multiply †		IND	IND	IND
Xj	−0	−0	0			IND	IND	IND
	+ ∞	+ ∞	− ∞	IND	IND	+ ∞	− ∞	IND
	− ∞	− ∞	+ ∞	IND	IND	− ∞	+ ∞	IND
	±IND	IND	IND	IND	IND	IND	IND	IND

†If both operands used in the integer multiply are normalized, an underflow results.

Table 5-5. Xj Divided by Xk (44, 45 Instructions)

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

Normalized Numbers

A normalized floating-point number has as large a coefficient and as small an exponent as possible. A floating-point number in packed format is normalized if the coefficient sign bit is different from bit 47. This condition indicates that the coefficient has been left-shifted until bit 47 contains the most-significant bit in the coefficient; therefore, the floating-point number has no leading sign bits in the coefficient. The normalized instructions perform the coefficient shift. The floating-multiply and floating-divide instructions deliver normalized results when provided with normalized operands. The floating-add instructions may deliver unnormalized results even when both operands are normalized. Therefore, it is necessary to perform the normalize operation after each sequence of floating-add or floating-subtract operations if the result is to be kept in a normalized form.

Rounding

Floating-point instructions round the results in single-precision computation. These instructions execute in the same amount of time as the unrounded versions. The operands are modified to accomplish the rounding function. The amount of bias introduced by the rounding operation varies and is affected by the coefficient value in the operands. The descriptions of the round instructions define the effects of rounding in detail.

Double-Precision Results

The floating-point arithmetic instructions generate double-precision results. Use of unrounded instructions allows separate recovery of upper- and lower-half results with proper exponents. Rounded instructions allow only upper-half results to be obtained. Two instructions, one single-precision and one double-precision, are required to retrieve an entire double-precision result.

To add or subtract two floating-point numbers, the coefficient with the smaller exponent enters the upper half of an accumulator and is right-shifted by the difference of the exponents. The other coefficient is then added into the upper half of the accumulator. The result is a double-length register (figure 5-3).

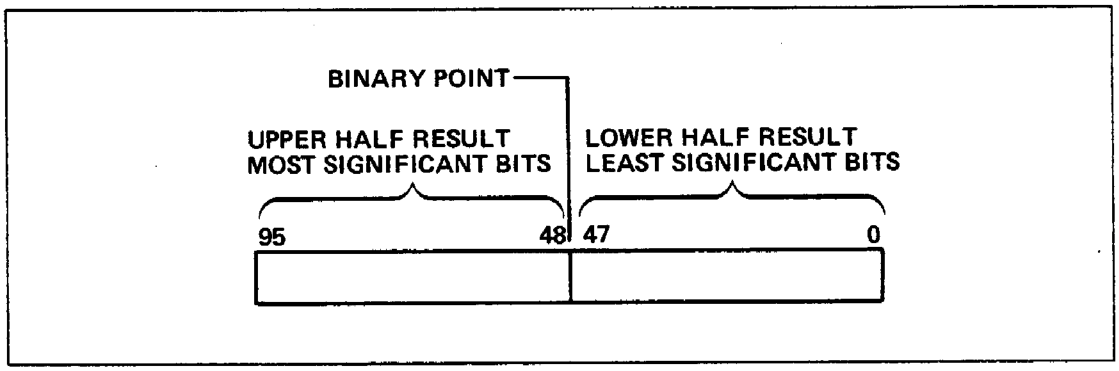


Figure 5-3. Floating-Add Result Format

If single precision is selected, the upper 48 bits of the 96-bit result and the larger exponent are returned as the result. Selecting double precision causes only the lower 48 bits of the 96-bit result and the larger exponent minus 60 (octal) to be returned as the result. The subtraction of 60 (octal) is necessary because the binary point is effectively moved from the right of bit 48 to the right of bit 0. A 96-bit product generates from two 48-bit coefficients. The result of a multiply is a double-length register (figure 5-4).

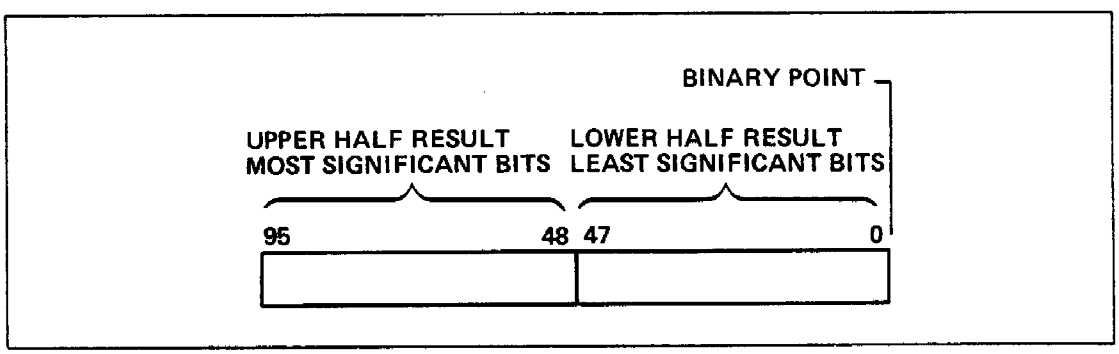


Figure 5-4. Multiply Result Format

If single precision is selected, the upper 48 bits of the product and the sum of the exponents plus 60 (octal) are returned as the result. The addition of 60 (octal) is necessary because the binary point effectively moves from the right of bit 0 to the right of bit 48 when the upper half of the 96-bit result is selected. If double precision is selected, the result is the lower 48 bits of the product and the sum of the exponents.

Fixed-Point Arithmetic

Fixed-point addition and subtraction of 60-bit numbers are handled by the long-add instructions (36 and 37). Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 59), and the binary point is to the right of the low-order bit position (bit 0).

The increment instructions (50 through 77) handle fixed-point addition and subtraction of 18-bit numbers. Negative numbers are represented in one's complement notation, and overflows are ignored. The sign bit is in the high-order bit position (bit 17), and the binary point is to the right of the low-order position (bit 0).

Integer multiplication is handled as a subset operation of the floating-multiply (42) instruction. The integer multiply requires that both 47-bit integer operands have zero exponents and are not normalized. The result is 48 bits with sign extension. Normalized operands cause underflow results to be reported. If the results exceed 48 bits, overflow is not detected.

An integer divide takes several steps. For example, an integer quotient X1 equal to X2/X3 is produced by the following steps.

Instructions	Remarks
1. Pack X2 from X2 and B0	Pack X2
2. Pack X3 from X3 and B0	Pack X3
3. Normalize X3 in X0 and B0	Normalize X3 (divisor)
4. Normalize X2 in X2 and B0	Normalize X2 (dividend)
5. Floating quotient of X2 and X0 to Xi	Divide
6. Unpack X1 to X1 and B7	Unpack quotient
7. Shift X1 nominally left B7 places	Shift to integer position

The divide requires that both integer (2^{47} maximum) operands must be in floating-point format, and the dividend coefficient must be less than two times the divisor coefficient. The normalize X3 instruction ensures this condition.

The normalize X3 instruction left-shifts the divisor n places ($n \geq 0$), providing a divisor exponent of negative n. The quotient exponent is then ≤ 0 minus (-n) minus 48 equals n minus 48<0.

After unpacking and left-shifting nominally, the negative (or zero) value in B7 right-shifts the quotient 48 minus n places, producing an integer quotient in X1. A remainder may be obtained by an integer multiply of X1 and X3 and subtracting the result from X2.

Integer Arithmetic

Integer divide packs the integers into floating-point format, using the pack instruction with a zero-exponent value.

In integer multiplication, a 48-bit product can be formed by using the double-precision multiply instruction. Both operands must have an exponent value of ± 0, and the coefficients cannot both be normalized. The result is sign-extended to 60 bits and sent to an X register.

In integer division, the divisor must be normalized, but the dividend does not have to be normalized. The resulting quotient must be unpacked and the coefficient must be shifted by the amount of the unpacked exponent using the left-shift (22) instruction to obtain the integer quotient.

Compare/Move Arithmetic

The compare/move arithmetic provides multiple-character manipulation. The characters are 6 bits long. Characters can be moved from one CM location to another, and fields of characters can be compared either directly or through a collate table.

The move direct instruction moves a field of up to 127 characters from one location to another location as specified in the instruction. The move indirect instruction performs the same kind of move, but a CM reference is used to obtain the parameters. The move indirect instruction moves a field of up to 8181 characters.

The compare collated instruction compares two fields of up to 127 characters. When two characters are unequal, the characters are referenced in a collate table, and the values are compared. If those values are unequal, the field with the larger character is indicated. The compare uncollated instruction compares two fields of up to 127 characters and indicates the larger of the first character pair that is found to be unequal.

CMU instructions are provided for compatibility with previous systems. For better performance, recompile jobs to avoid use of CMU instructions.

Instruction Lookahead Purge Control

Prefetching of instructions at a branch target address by instruction lookahead hardware can lead to program failures if a program modifies its own code dynamically. Under normal conditions, the lookahead registers are purged by execution of a return jump instruction (010), UEM read instruction (011), exchange jump instruction (013), or unconditional branch instruction (02). Selecting extended purge control extends these conditions. When extended purge control is in effect, lookahead registers are also purged by execution of any conditional jump instruction (03 through 07) or any CM store instruction (50 through 57 when i equals 6 or 7). To enable extended purge control, the system sets bit 52 of the flag register in the CYBER 170 exchange package. When self-modifying code is present, it may be helpful to set extended purge control; however, the additional purging causes a degradation in execution and does not cover all cases of code modification.

Purge Control

If normal purge conditions are in effect, a store instruction that modifies a sequential instruction must modify at least P plus 6 words ahead to ensure execution of the modified code. In addition, a store instruction followed by a branch to a modified instruction executes the modified code only if there are at least 12 executed instructions between the store and the modified code.

If the extended purge option is selected, a store instruction can modify the next sequential instruction and be assured of executing the modified instruction. Likewise, a store instruction followed by a branch to a modified instruction always executes the modified code.

Error Response

When the CP detects or is informed of an error, it records the error. Depending on the type of error and the exit mode selection bits set in the EM register, the program in execution may be interrupted. If the error is an illegal instruction or an address-range error on an RNI or branch, the program interruption is unconditional. For other types of errors, the exit mode selection bits determine whether or not the program is interrupted. If the exit mode selection bit is set and the corresponding condition is detected, the program is interrupted. The exit mode selection bits are contained in word N plus 3 of the exchange package. Figure 5-5 shows the format of the exit condition register at (RAC). Table 5-6 describes the possible contents of the register. Tables 5-7 and 5-8 list CP error responses.

The CP has the following error conditions: illegal instructions, hardware errors, and conditional software errors.

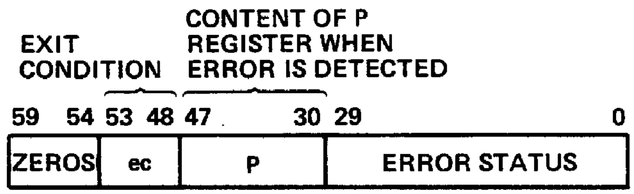


Figure 5-5. Format of Exit Condition Register at (RAC)

Table 5-6. Contents of Exit Condition Register at (RAC)

Field	Description
ec	6-bit exit condition code: **Code** — **Condition** 00_8 — Illegal instruction. 01_8 — Address-range error (bit 48). 02_8 — Floating-point infinite (bit 49). 04_8 — Floating-point indefinite (bit 50). 20_8 — Processor-detected malfunction. 67_8 — Hardware malfunction.
P	When an error exit occurs, the content of the P register may not correspond to the address of the instruction that caused the error exit. The P register may have been incremented prior to the execution of the instruction.
ERROR STATUS	Nonzero information in bits 0 through 29 is error status for customer engineering and maintenance.

Table 5-7. Error Exits in CYBER 170 Monitor Mode (MF=1)

Error Condition	Error Response: Exit Mode Selected	Error Response: Exit Mode Not Selected
Illegal instruction or 00 instruction.	1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Interrupt to executive state. 4. CP stops in executive state.	1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state.	1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Interrupt to executive state. 5. CP stops in executive state.	1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Interrupt to executive state. 4. CP stops in executive state.	1. N/A (exit mode is always selected regardless of status of EM register bit 48).

(Continued)

Table 5-7. Error Exits in CYBER 170 Monitor Mode (MF=1) (Continued)

Error Condition	Error Response: Exit Mode Selected	Error Response: Exit Mode Not Selected
Exit condition bit 48 set on CMU instruction. 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range.	1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Store P and exit bits (01) at RAC. 4. CP stops in executive state.	1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.	1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. 3. Interrupt to executive state. 4. CP stops in executive state.	1. Execute instruction as a pass. 2. Exit to next 60-bit word and continue execution.
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.	1. Execute instruction as a pass. 2. Store P and exit condition bits (01) at RAC. P equals address of following instruction. 3. Interrupt to executive state. 4. CP stops in executive state.	1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.	1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Interrupt to executive state. 3. CP stops in executive state.	1. Continue execution.
Any hardware parity error or double SECDED error.	1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. CP stops in executive state.	1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. CP stops in executive state.

Table 5-8. Error Exits in CYBER 170 Job Mode (MF=0)

Error Condition	Error Response: Exit Mode Selected	Error Response: Exit Mode Not Selected
Illegal instruction or 00 instruction.	1. The instruction is not executed. 2. Store P and exit condition bits (00) at location RAC. P equals address of illegal instruction. 3. Exchange jump to MA and set CYBER 170 MF.	1. N/A (exit mode is always selected).
Exit condition bit 48 set by an incremental read with an address out of range (AOR).	1. The X register is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of increment instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF.	1. Inhibit read, X unchanged. 2. Continue execution.
Exit condition bit 48 set by an incremental write with an address out of range (AOR).	1. Block write operation; content of CM is unchanged. 2. The A register contains the AOR address. 3. Store P and exit condition bits (01) at location RAC. P equals address of instruction or address of instruction following the increment. 4. Exchange jump to MA and set CYBER 170 MF.	1. Inhibit write, CM unchanged. 2. Continue execution.
Exit condition bit 48 set by an RNI or branch address out of range.	1. Inhibit execution. 2. Store P and exit condition bits (01) at location RAC. P equals address of instruction required by RNI or address of branch destination instruction. 3. Exchange jump to MA and set CYBER 170 MF.	1. N/A (exit mode is always selected regardless of status of EM register bit 48).

(Continued)

Table 5-8. Error Exits in CYBER 170 Job Mode (MF=0) (Continued)

Error Condition	Error Response: Exit Mode Selected	Error Response: Exit Mode Not Selected
Exit condition bit 48 set on CMU instruction. 1. C1 or C2 greater than 9. 2. K1 or K2 address out of range.	1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Conditon 2 causes the instruction to go unexecuted. 3. Store P and exit bits (01) at RAC. 4. Exchange jump to MA and set CYBER 170 MF.	1. Detected by executive state during the execution of compare/move instruction. 2. Condition 1 omits reading/writing; CM is unchanged. Condition 2 causes the instruction to go unexecuted. 3. Continue with next instruction.
Exit condition bit 48 set by a UEM address range check for instructions 011 and 012.	1. Execute instruction as a pass. 2. Store P and exit bits (01) at RAC. continue execution. 3. Exchange jump to MA and set CYBER 170 MF.	1. Execute instruction as a pass. 2. Exit to next 60-bit word and
Exit condition bit 48 set by a UEM address range check for instructions 014 and 015.	1. Execute instruction as a pass. 2. Stop CP. 3. Store P and exit condition bits (01) at location RAC. 4. Exchange jump to MA and set CYBER 170 MF.	1. Execute instruction as a pass. 2. Exit to next parcel and continue execution.
Exit condition bit 49 set by infinite condition, or bit 50 set by indefinite condition.	1. Store P and exit condition bits (02 for infinite or 04 for indefinite). P equals address of arithmetic instruction or address of instruction following. 2. Exchange jump to MA and set CYBER 170 MF.	1. Continue execution.
Any hardware parity error or double SECDED error.	1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.	1. Interrupt to executive state. 2. Executive state stores P and exit condition bits (20) at RAC. 3. Exchange jump to MA and set CYBER 170 MF.

Illegal Instructions

An instruction is illegal when it has an illegal operating code, an illegal operating parameter, or when it is positioned so that it begins in one instruction word and extends into the next instruction word. In the CYBER 170 job mode, illegal instructions cause an exchange to the CYBER 170 monitor mode. In the CYBER 170 monitor mode, illegal instructions cause a jump to executive state. The CP stops. CP illegal instructions are:

- 017.
- 011, 012, 013, 464, 465, 466, 467 if they do not begin at parcel 0.
- 011, 012, 014, 015 if the UEM enable flag in the flag register of the CYBER 170 exchange package is clear.
- Any 30-bit instruction that begins at parcel 3.

Hardware Errors

CP/CM hardware errors are: data parity errors, address parity errors, and double-bit errors. If the CP is in CYBER 170 job mode, a hardware error causes a jump to executive state, which returns to CYBER 170 monitor mode. If the CP is in CYBER 170 monitor mode, a hardware error causes a jump to executive state. The CP halts. The instruction being executed when such a fault is detected is not necessarily connected with the fault.

Conditional Software Errors

Conditional software errors are caused by address-range errors and floating-point infinite/indefinite operands or results. A conditional software error causes action, depending on bits set in the EM field in the current CYBER 170 exchange package. If the bit reserved for use with the specific type of error is clear, the error is ignored in both CYBER 170 job and CYBER 170 monitor modes. If the bit is set and the error occurs in the CYBER 170 job mode, it causes an exchange to the CYBER 170 monitor mode.

If the bit is set and the error occurs in the CYBER 170 monitor mode, it causes an interrupt to executive state.

Memory Programming

All references to CM by the CP for instructions or read/write data are made relative to RAC. The RAC defines the lower limit of the addresses of a program in CM. The upper limit of the program addresses is defined by FLC added to RAC.

All references to UEM by the CP for instructions or read/write data are made relative to RAE. The RAE defines the lower limit of the addresses of a program/data in UEM. The upper limit of the addresses is defined by FLE added to RAE.

The field length is a number of 60-bit words established by the operating system prior to program execution. All references to CM or UEM for a program/data must be within the field established for that program.

During a CYBER 170 exchange jump, RAC and FLC are loaded into respective registers to define the CM limits of the program that is initiated by the CYBER 170 exchange jump. RAE and FLE are loaded to define the UEM limits of a program.

Figure 5-6 shows the absolute and relative memory addresses, RAC, FLC, RAE, and FLE register relationships. For a program to operate within the established limits, the following conditions must exist.

- For absolute memory addresses:

  $RAC \leq (RAC + P) < (RAC + FLC)$

- For relative memory addresses:

  $0 \leq P < FLC$

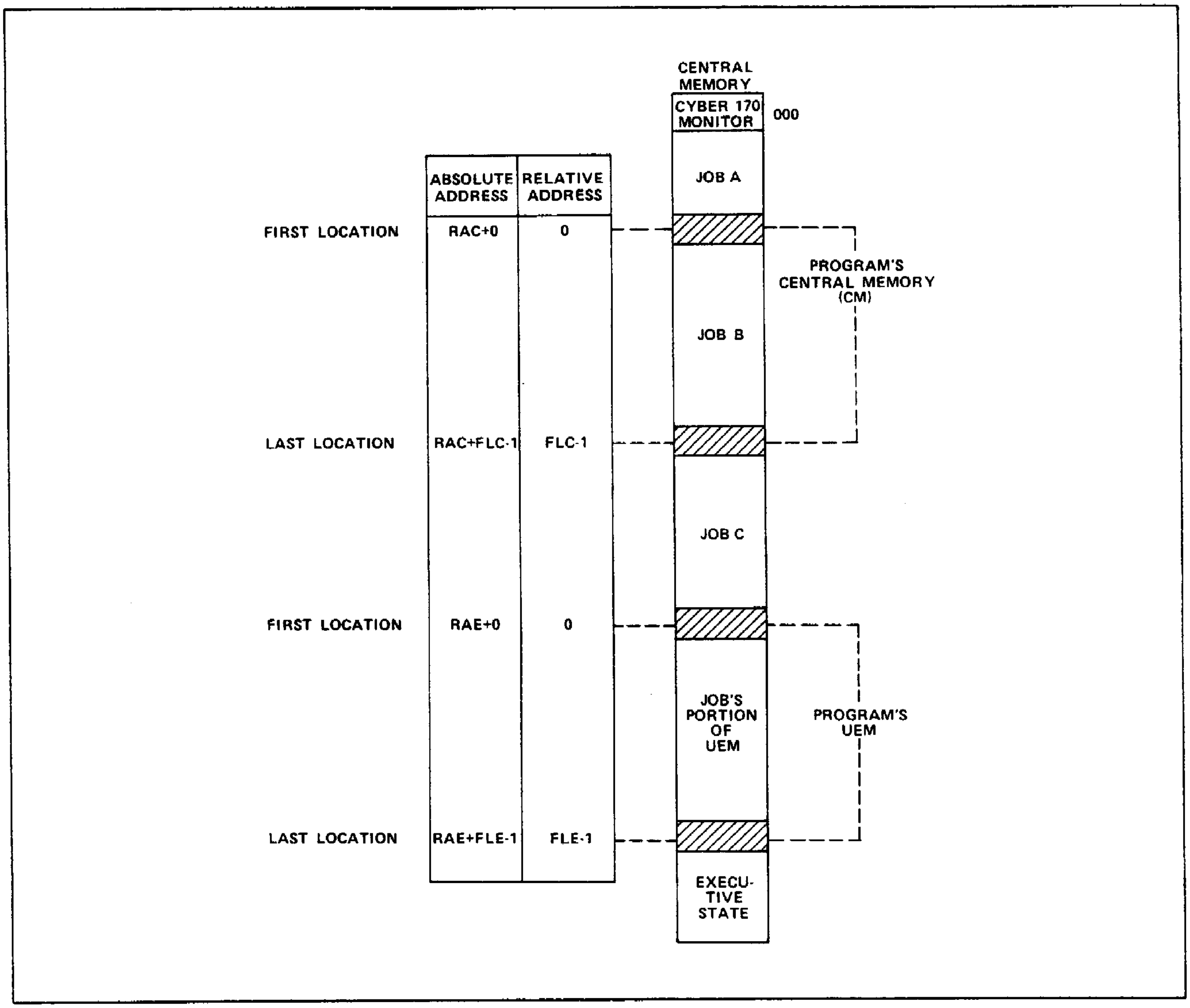


Figure 5-6. Memory Map

Addressing Modes

UEM can be used in either of two addressing modes: standard or expanded. Standard addressing mode provides addressing up to 21 bits in a 24-bit format. Expanded addressing mode provides addressing up to 24 bits in a 30-bit format. Addressing mode is determined by the expanded addressing select flag, bit 55 of word 3, in the CYBER 170 exchange package.

Direct Read/Write Instructions (014, 015, 660, 670)

These instructions transfer one 60-bit word between the selected X register and a memory location, using a 21-bit relative address. Instructions 660 and 670 use the memory address Xk (21 bits) plus RAC (21 bits) to address CM. Instructions 014 and 015 use the memory address Xk (21 bits) plus RAE (21 bits) to address UEM.

Block Copy Instructions (011, 012)

These instructions transfer up to 131 071 60-bit words between fields in CM and UEM. The UEM address is X0 plus RAE (bits 0 through 22 in standard addressing mode; bits 0 through 28 in expanded addressing mode). The CM address is A0 plus RAC (if the block copy flag is clear in the CYBER 170 exchange package) or X0 (bits 30 through 50) plus RAC (if the block copy flag is set).

The transfers occur in blocks of up to 64 words, during which other CP activities are suspended.

These instructions are 30-bit instructions that must start at parcel 0. If the UEM address has bit 21 or bit 22 set in standard addressing mode (bit 28 if in expanded addressing mode), 0's are transferred to CM and the next instruction is taken from parcel 2 of the same instruction word. If this is not the case on a block read, the next instruction is taken from parcel 0 of the next instruction word. A transfer of all 0's can be made to central memory using the 011 instruction and setting bit 21 or 22 (or bit 28) of the address (X0 + RAE) when FLE is sufficiently large.

PP Programming

The PPs have access to all CM storage locations. One 64-bit word or a block of 64-bit words can be transferred from a peripheral processor memory (PPM) to CM or from CM to PPM. (Five 12-bit PP words equal one 64-bit CM word, with the leftmost 4 bits undefined.) Data from external devices is read into a PPM, and with additional instructions, is transferred to CM. Conversely, data is transferred from CM to a PPM and is then transferred by additional instructions to external devices. Addresses sent to CM from PPs are absolute or relocation addresses.

Central Memory Addressing by PPs

PPs address central memory using either absolute or relocation addressing. Every PP can read all central memory locations without restriction. Every PP has write access to central memory. The bounds register in central memory may also be set to limit write access from the IOU.

Instructions 24/25 load/store the relocation (R) register. If bit 17 of the A register is 0, bits 0 through 16 of A specify an absolute central memory address 0 through 377 777_8. If bit 17 of A is 1, bits 0 through 16 of A are added to the 28-bit R register to specify an absolute central memory address 0 through 0 007 777 777_8. If bit 17 of A changes during a transfer, the addressing mode also changes accordingly. The leftmost 7 bits of R represent unused extra addressing capacity. The rightmost 6 bits of R are appended 0's. Instruction 24 loads R from two consecutive PP memory locations. Instruction 25 stores R into two PP memory locations. Figure 4-4 shows how R is stored in PP memory.

PP Memory Addressing by PPs

PP instructions use 6-bit or 18-bit direct operands or access PP memory through direct, indirect, or indexed addressing.

Direct 6-Bit Operand

PP instructions in this category are no-address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct operand, zero-extended to 18 bits in calculations.

Direct 18-Bit Operand

PP instructions in this category are constant address instructions. They have the format OPCODEdm. The combined d and m fields are used as an 18-bit operand.

Direct 6-Bit Address

PP instructions in this category are direct-address instructions. They have the format OPCODEd. The d field is used as a 6-bit direct address, accessing PP memory locations 0 to 77_8.

Direct 12-Bit Address

PP instructions in this category are indexed direct-address instructions with zero index. They have the format OPCODEdm where d equals 0. The m field is used as a 12-bit direct address that accesses PP memory locations 0 through 7777_8.

Indexed 12-Bit Address

PP instructions in this category are indexed direct-address instructions. They have the format OPCODEdm where d equals 0. The m field is used as a 12-bit direct address (base address). The d field specifies a PP memory location from 1 to 77_8, the contents of which is a 12-bit one's complement number index. The indexed direct address is formed by adding the index to the base address as signed one's complement numbers. Overflow is ignored. When m plus (d) equals 7777, the result is set to 0000, except as follows: adding 7777 plus 7777 equals 7777. In general, adding 0000 or 7777 leaves the other number unchanged, except when the other number is also 0000 or 7777.

Indirect 6-Bit Address

PP instructions in this category are indirect-address instructions. They have the format OPCODEd. The 6-bit d field is used to read a 12-bit number from PP locations 0 through 77_8. This number is used as a 12-bit address to access PP memory locations 0 through 7777_8.

Central Memory Read/Write Instructions

PP instructions can read and write to central memory either single words or blocks of words.

PP Central Memory Read Instructions (60, 61)

Instruction 60 transfers one CM word into five 12-bit PP memory words. Instruction 61 transfers a block of 1 through 811 CM words into 5 through 4095 12-bit PP words. It is possible to transfer up to 4096 CM words overwriting PP memory cyclically; location 0, however, has special properties. The Central Read description in chapter 4 has more information on instruction 61.

PP Central Memory Write Instructions (62, 63)

Instruction 62 transfers five 12-bit PP memory words into 1 CM word. Instruction 63 transfers 5 through 4095 PP memory words into 1 through 811 CM words. It is possible to transfer up to 20 480 PP memory words, repeating information from PP memory cyclically.

Input/Output Channel Communications

Data transfers to and from external devices are controlled by PP instructions 64 through 77. The assignment of PPs, transfer priorities, and resolution of conflicts are software responsibilities.

Channel parity and reservation must be provided for, using the channel marker flag and/or software interlocks in central memory. After any conflicts have been resolved, proceed as follows:

Action	Typical Instruction
1. Clear the error flag.	Jump if the error flag is set, and clear the flag (661).
2. Verify inactive status.	Jump if active (640).
3. Verify read status:	
Prepare for reading the summary status.	Function m (77).
Verify that the device responded.	Jump if active (640).
Activate the channel.	Activate (74).
Read the summary status.	Input to A (70).
Verify the error flag is clear.	Jump if the error flag is set (661).
Analyze the summary status.	Logical product (12). Zero jump (04).
4. Enter the number of words to A.	Load d (14).
5. Prepare for input/output:	
Verify inactive status.	Jump if active (640).
Prepare for read/write.	Function m (77).
Verify that the device responded.	Jump if active (640).

Action	Typical Instruction
6. Read/write data:	
Activate the channel.	Activate (74).
Read/write data.	Input/output A words (71/73).
If write, loop until empty.	Jump if full (660).
Disconnect the channel.	Deactivate (75).
Verify inactive status.	Jump if active (640).
7. Verify transfer integrity:	
Verify A words were transferred (refer to note).	Nonzero jump (05).
Verify the error flag is clear.	Jump if error flag set (661).
Verify inactive status.	Jump if active (640).
Prepare for reading device status.	Function m (77).
Verify that the device responded.	Jump if active (640).
Activate the channel.	Activate (74).
Read the device status.	Input to A (70).
Verify the error flag is clear.	Jump if error flag set (661).
Analyze device status.	Logical product (12). Nonzero jump (05).
Disconnect the channel.	Deactivate (75).

NOTES

If A equals the original value, no words were transferred.

If A is not equal to 0, the device or another PP ended the transfer.

Inter-PP Communications

Any PP can communicate with any other PP using any channel (except the real-time clock) by omitting the conditioning of the external devices of that channel for a data transfer. Both single-word and block transfers can be used. Either the sending or the receiving PP can activate the channel used, after which the sending PP outputs data into the channel register of the channel concerned and the receiving PP inputs data from the same register. The transfer rate is 1 word every 250 ns, except when the transfer is between PPs in different barrels but in the same time slot. In such a case, the transfer rate is 1 word every 500 ns. PPs that use the same time slots are as follows:

Slot	PP Number
1	0, 5, 20, 25
2	1, 6, 21, 26
3	2, 7, 22, 27
4	3, 10, 23, 30
5	4, 11, 24, 31

Software resolves priority and reservation problems arising in inter-PP communications by interlocks stored in CM or by other means.

PP Program Timing Considerations

Some external equipment may require timing considerations in issuing function, activate, and input instructions. Refer to the applicable external equipment reference manual. Such timing considerations may, for example, be required to ensure that the equipment attains a proper speed before data is sent (required by some magnetic tape equipment). Also, equipment that terminates a data transfer by resetting the active flag to inactive often requires timing considerations in issuing the next function instruction.

Channel Operation

Channel Control Flags

Channel operation is affected by the channel active/inactive and full/empty flags and, depending on the status of these two flags, the channel is said to be active, inactive, full, or empty. Each channel also has a marker flag for software use and an error flag for indicating transmission parity errors.

Channel Active/Inactive Flag

A channel is normally activated by a function (76 or 77) instruction or by an activate channel (74) instruction. An external device can also activate the channel.

A function instruction conditions the external device for a coming data or status information transfer. The instruction places a 12-bit function word plus parity in the channel register and sets the active and full flags. The function word and a function signal are sent to the external device. No active or full signals are sent during a function instruction. The external device accepts the function word and sends an inactive signal, which clears the channel active and full flags, clearing the channel register.

An activate channel instruction prepares a channel for data transfer and sends an active signal to the external device. Subsequent input or output instructions transfer data. A disconnect channel (75) instruction after a data transfer returns the channel to an inactive state, and an inactive signal is sent to the external device.

Register Full/Empty Flag

A register is full when it contains a function or data word for an external device or contains a word received from the external device. The register is empty when the flag clears. The flag is turned on or off as the register changes state. A channel can only be full when it is active.

On data output, the processor places a word in the channel register (the channel should be active and empty) and sets a full flag. The data word plus parity and a full signal are sent to the external device. The external device accepts the word and sends an empty signal to the channel, which clears the full flag, clearing the channel register. The active and empty status of the channel signals the PP to send the next word to the register.

On data input, the external device sends a word and a full signal to the data channel. The word is placed in the channel register, and the full flag sets. The PP stores the word and clears the full flag, clearing the data register. An empty signal is sent to the external device, signaling it to send the next data word.

Channel (Marker) Flag Instructions (641, 651)

Software uses this flag software as a marker. This flag does not affect hardware operation. When PPs in the same time slot use this flag, priority conflicts exist. For channel 17_8 (maintenance channel) marker flag, hardware resolves priority problems. For other channels, software must resolve such conflicts. Any five consecutively numbered PPs are not in the same time slot.

Error Flag Instructions (661, 671)

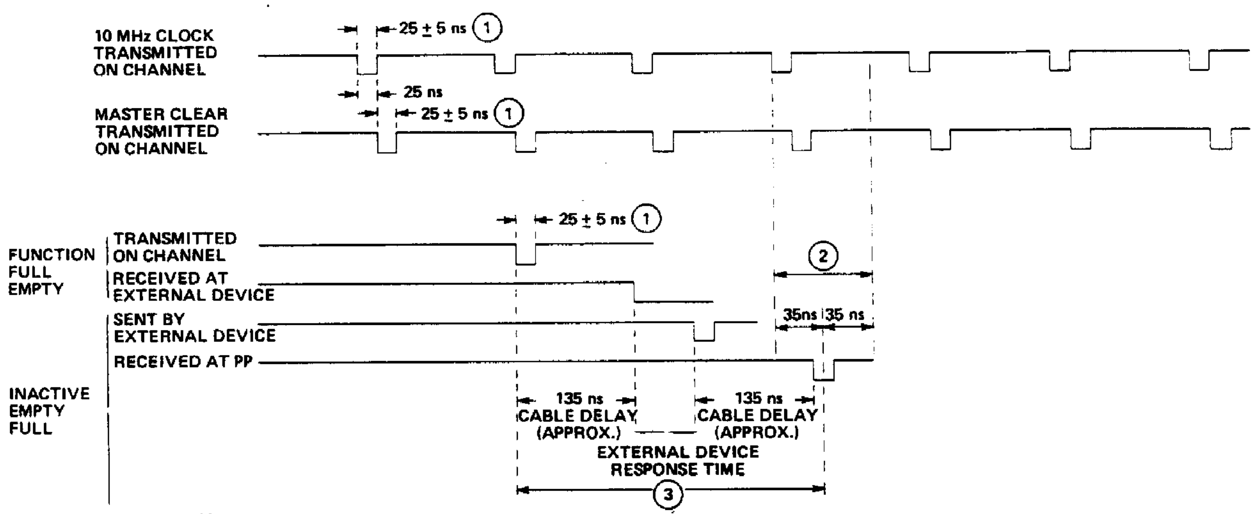
This flag indicates an input data parity error on the specific channel being tested. The flag also indicates an output data parity error on channels that have the capability of sending an error signal to the IOU in case of such an error. The status register of the device concerned must be read to verify output data integrity.

Channel Transfer Timing

Figure 5-7 shows channel transfer timing. All signal pulses are 25 ± 5 ns in width and occur 25 ± 5 ns following the 10-MHz clock.

To maintain the fastest possible cycle time (500 ns), a function/full/empty pulse from the PP must be answered with an inactive/empty/full pulse, respectively, within 310 ± 35 ns. If the maximum speed is not required, this response time may be increased by multiples of 100 ns.

The PP master clock frequency can be varied by ± 2 percent. The peripheral devices used must tolerate this frequency variation.



NOTES:

1. ALL TRANSMISSION PULSE WIDTHS (INCLUDING DATA, FULL, EMPTY, ETC.) ARE 25 ± 5 ns
2. TO AVOID LOST DATA, ALL INPUTS FROM THE CHANNEL TO THE PP MUST ARRIVE WITHIN THE 70 ns. INPUTS MAY BE EARLIER OR LATER BY 100 ns MULTIPLES.
3. TOTAL TURNAROUND TIME BETWEEN FUNCTION AND INACTIVE IS MEASURED AT PP. THIS TIME VARIES DUE TO EXTERNAL DEVICE RESPONSE TIME BUT MUST BE WITHIN 310 ± 35 ns TO MAINTAIN THE 500 ns CYCLE TIME.

Figure 5-7. Channel Transfer Timing

Input/Output Transfers

The following paragraphs discuss input/output transfers with the PP.

Data Input Sequence

The external device sends data (figure 5-8) to the PP via the controller as follows:

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. At the same time, the PP sends the first of a group of words and function signals to all controllers. The function signals cause all controllers to sample the words and identify the words as function codes rather than data words. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.

2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn, drops the full flag and clears the channel register.

3. The PP sets the channel active flag and sends an active signal to the controller, which signals the input equipment to start sending data.

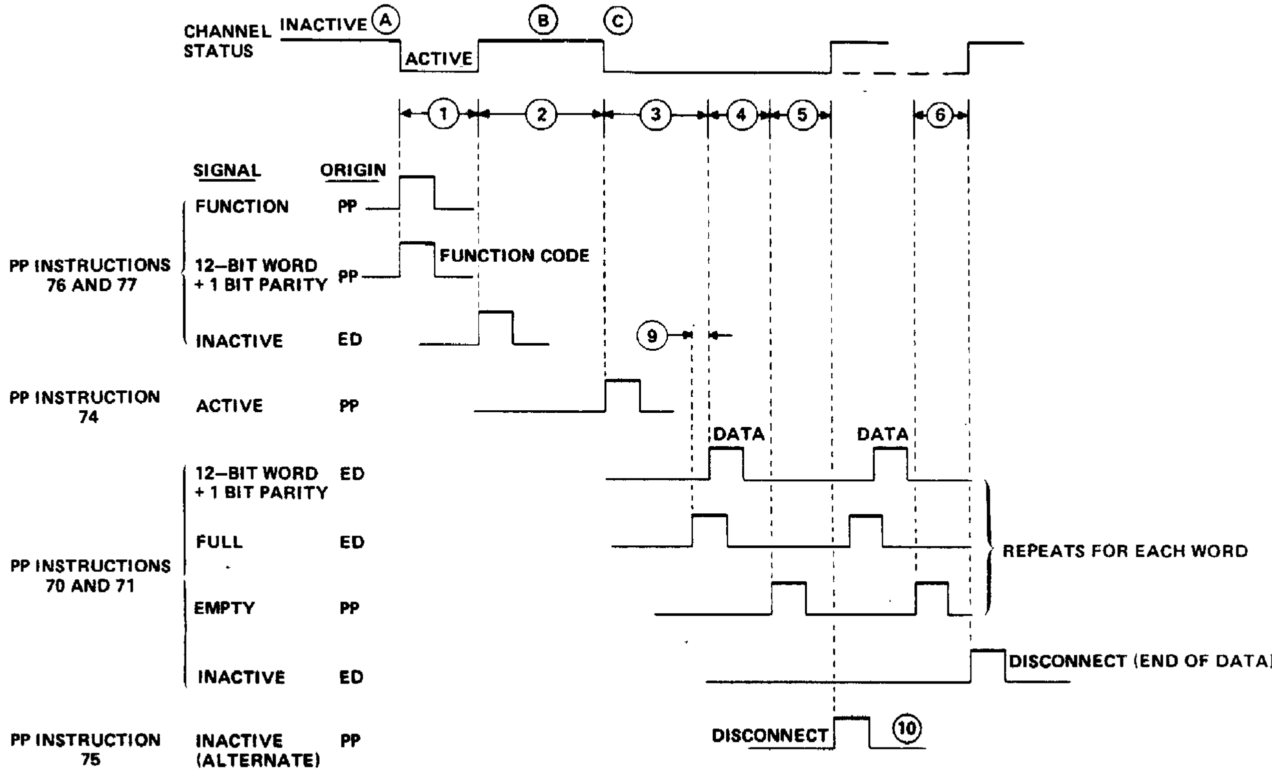
4. The input equipment reads a 12-bit data word plus 1 parity bit and then sends the word with parity to the channel register with a full signal, which sets the channel full flag (10 to 15 nanoseconds after the data arrives).

5. The PP stores the word, drops the full flag, and returns an empty signal, indicating acceptance of the word. The input equipment clears its data register and prepares to send the next word.

6. Steps 4 and 5 repeat for each word transferred.

7. At the end of the transfer, the controller clears its active condition and sends an inactive signal to the PP to indicate the end of the data. The signal clears the channel active flag to disconnect the controller and the PP from the channel.

8. As an alternative, the PP may choose to disconnect from the channel before the input equipment has sent all its data. The PP does this by dropping the active flag and sending an inactive signal to the controller, which immediately clears its active condition and sends no more data, although the input equipment may continue to the end of its record or cycle (for example, a magnetic tape unit would continue to end-of-record and stop in the record gap).



NOTES:

1. TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.
2. TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.
3. TIME IS A FUNCTION OF ED.
4. TIME IS A FUNCTION OF PP. MINIMUM TIME IS 1 MINOR CYCLE. MAXIMUM TIME IS UP TO 4 MINOR CYCLES TO ALLOW OPERATION WITHIN 1 MAJOR CYCLE.
5. TIME IS A FUNCTION OF PP. MINIMUM TIME IS 2 MAJOR CYCLES. MAXIMUM TIME IS AN INTEGRAL MULTIPLE OF MAJOR CYCLES.
6. TIME IS A FUNCTION OF ED.
7. MAJOR CYCLE TIME IS 250 NS.
8. MINOR CYCLE IS 50 NS.
9. TIME IS A FUNCTION OF ED. FULL SHOULD PROCEED THE DATA BY A MINIMUM OF 5 NS (15 NS MAXIMUM) TO REMOVE THE CLEAR ON THE INPUT DATA RECEIVERS.
10. PP MAY DISCONNECT AFTER EMPTY SIGNAL OF ANY ED WORD. STATUS REQUEST DISCONNECTS IN THIS MANNER.

(A) CHANNEL MUST BE PREVIOUSLY INACTIVE.

(B) CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

(C) CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-8. Data Input Sequence Timing

Data Output Sequence

The PP sends data (figure 5-9) to the external device as follows.

1. The PP places a function word in the channel register and sets the full flag and the channel active flag. The function signal causes all controllers to sample the word and identify the word as a function code rather than a data word. Connect codes select controllers and modes of operation and clear nonselected controllers. Only selected controllers are connected.

2. The controller sends an inactive signal to the PP, indicating acceptance of the function code. The signal drops the channel active flag, which in turn, drops the full flag and clears the channel register.

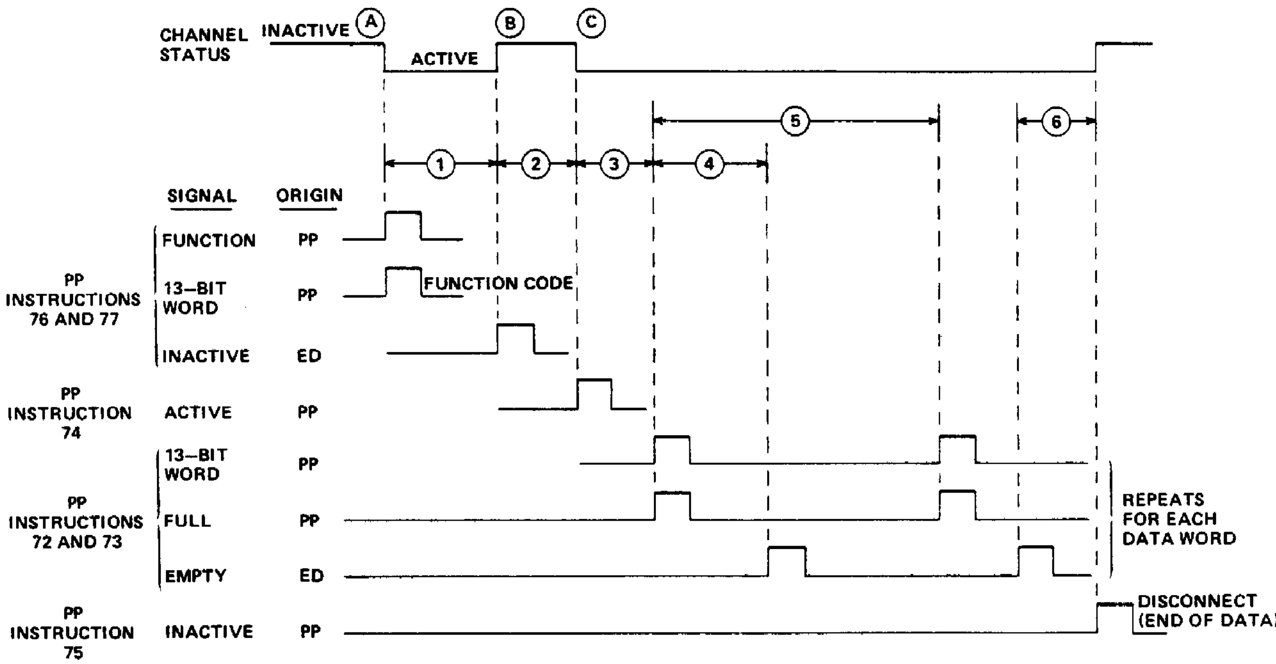
3. The PP sets the channel active flag and sends an active signal to the controller, which signals the output equipment that data flow is starting.

4. The PP places a 12-bit data word plus 1 parity bit in the channel register and sets the full flag. Coincidently, the PP sends a word with parity and a full signal to the controller.

5. The controller accepts the word and sends an empty signal to the PP where the signal clears the channel register and drops the full flag.

6. Steps 4 and 5 repeat for each PP word.

7. After the last word is transferred and acknowledged by the controller empty signal, the PP drops the channel active flag and turns off the controller with an inactive signal.



NOTES:

1. TIME IS A FUNCTION OF EXTERNAL DEVICE (ED). PP RECOGNIZES INACTIVE 1 MAJOR CYCLE (OR A MULTIPLE OF MAJOR CYCLES) AFTER FUNCTION. THE PP MUST PREVIOUSLY RECEIVE INACTIVE.

2. TIME IS A FUNCTION OF PERIPHERAL PROCESSOR (PP). MINIMUM TIME IS 1 MINOR CYCLE. ACTUAL TIME IS A FUNCTION OF THE PP PROGRAM.

3. TIME IS A FUNCTION OF ED.

4. TIME IS A FUNCTION OF PP. MINIMUM TIME IS 1 MINOR CYCLE. MAXIMUM TIME IS UP TO 4 MINOR CYCLES TO ALLOW OPERATION WITHIN 1 MAJOR CYCLE.

5. TIME IS A FUNCTION OF PP. MINIMUM TIME IS 2 MAJOR CYCLES. MAXIMUM TIME IS AN INTEGRAL MULTIPLE OF MAJOR CYCLES.

6. TIME IS A FUNCTION OF ED.

7. MAJOR CYCLE TIME IS 250 NS.

8. MINOR CYCLE IS 50 NS.

A. CHANNEL MUST BE PREVIOUSLY INACTIVE.

B. CHANNEL REMAINS ACTIVE UNTIL ED SENDS INACTIVE.

C. CHANNEL MUST BE PREVIOUSLY INACTIVE.

Figure 5-9. Data Output Sequence Timing

System Console Programming

Keyboard

A PP transmits function code 7020_8 to request data from the keyboard of the system console. The PP then activates the input channel and inputs one character from the keyboard. This character enters as the lower 6 bits of the word; the upper bits are cleared. There is no status report by the keyboard. Table 5-9 lists the keyboard character codes.

Data Display

Data is displayed within an 8- by 11-inch area of a cathode-ray tube (CRT). The display can be in character mode (alphanumeric) and/or dot mode (graphic). Two presentation areas (left and right) are displayed. Each is made up of 262 144 dot locations arranged in a 512- by 512-dot format. Each dot position is determined by the intersection of X and Y coordinates. The lower left corner dot is octal address X=6000 and Y=7000, and the upper right corner dot is octal address X=6777 and Y=7777. An optional CC 634B system console is available. Refer to the hardware reference manual listed in the preface for additional information regarding this terminal.

Character Mode

In character mode, three sizes are provided. Large characters are arranged in a 32- by 32-dot format with 16 characters per line. Medium characters are arranged in a 16- by 16-dot format with 32 characters per line. Small characters are arranged in an 8- by 8-dot format with 64 characters per line. Table 5-10 lists the display character codes.

Table 5-9. Keyboard Character Codes

Character	Code	Character	Code
No data	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(	51
O	17	)	52
P	20	Left blank key	53
Q	21	=	54
R	22	Right blank key	55
S	23	,	56
T	24	.	57
U	25	Carriage return	60
V	26	Backspace	61
W	27	Space	62
X	30		
Y	31		
Z	32		

Table 5-10. Display Character Codes

Character	Code	Character	Code
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(	51
O	17	)	52
P	20	Space	53
Q	21	=	54
R	22	Space	55
S	23	,	56
T	24	.	57
U	25		
V	26		
W	27		
X	30		
Y	31		
Z	32		

Dot Mode

In dot mode, display dots are positioned by the X and Y coordinates. The X coordinates position the dots horizontally. The Y coordinates position the dots vertically and unblank the CRT for each dot. A series of X and Y coordinates form horizontal lines. A single X coordinate and a series of Y coordinates form vertical lines.

Codes

A single function word is transmitted to select the presentation, mode, and character size (character mode only). Figure 5-10 illustrates the function word format. The word following the function word specifies the starting coordinates for the display (for either mode). Figure 5-11 illustrates the coordinate data word. In character mode, the words that follow are display character codes. Figure 5-12 illustrates the character data word.

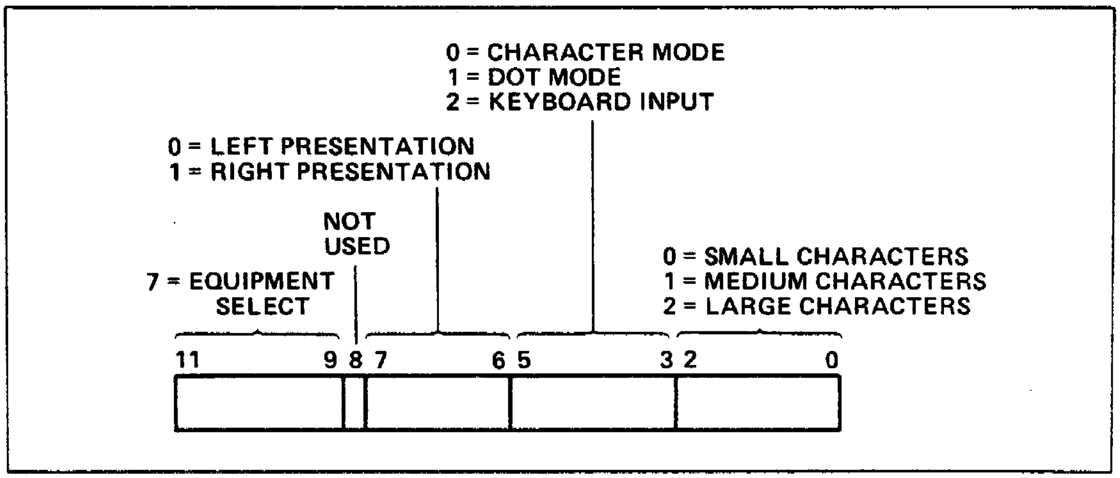


Figure 5-10. Display Station Output Function Code

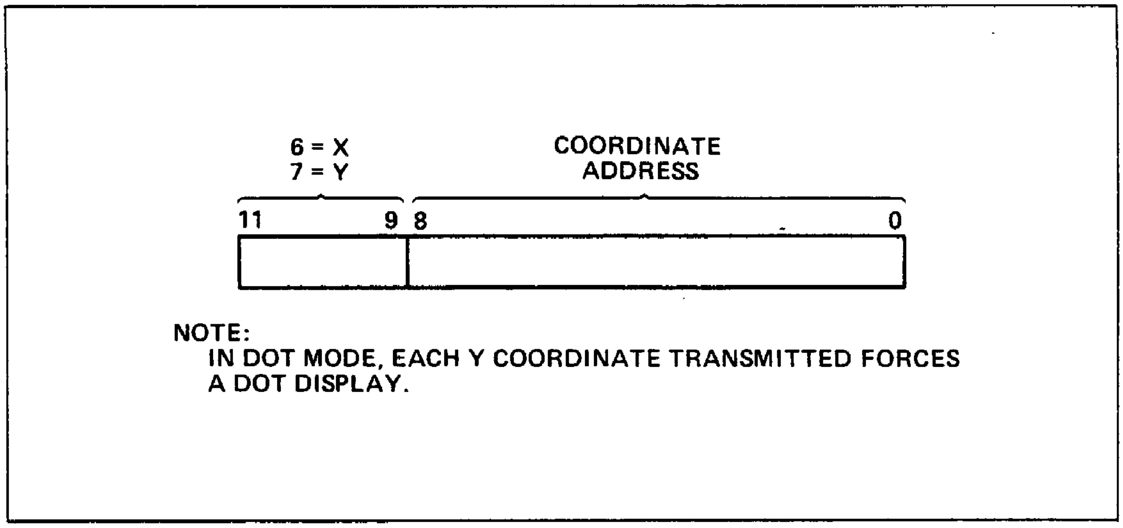


Figure 5-11. Coordinate Data Word

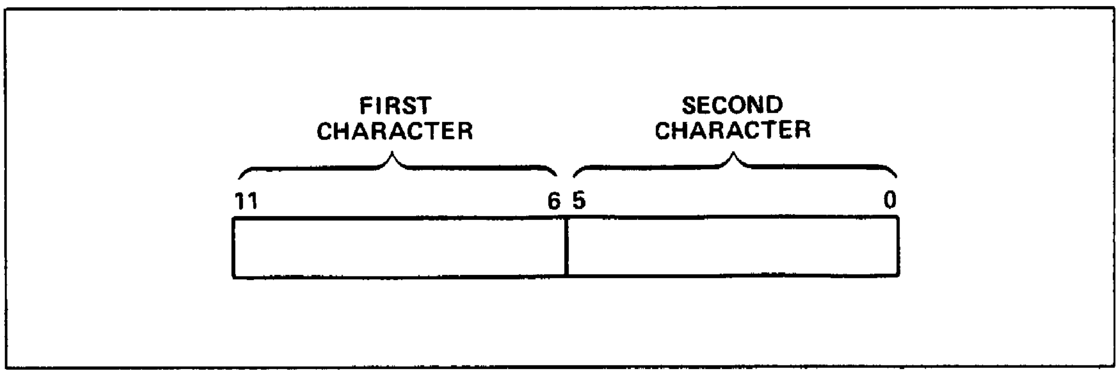


Figure 5-12. Character Data Word

When the display operation has started, the controller regulates character spacing on the line. A new coordinate data word must be sent to start each line. If new coordinates are not specified, data is written on the line specified by the active coordinate word, and information already on that line is overwritten. Character sizes can be mixed by sending a new function word and coordinate word for each size change. Spacing on a line can be varied by sending a coordinate word for the character that is to be spaced differently.

Programming Example

The following programming example (figure 5-13) requests an input of one line of data from the system console and displays this data on the CRT as it is being typed.

Programming Timing Considerations

When performing an output operation, the computer must wait at the end of the output for a channel-empty condition to prevent a loss of coordinates or data. A full jump at the end of the output ensures that the channel is empty and the display controller accepts the last word of the output before disconnecting from the channel.

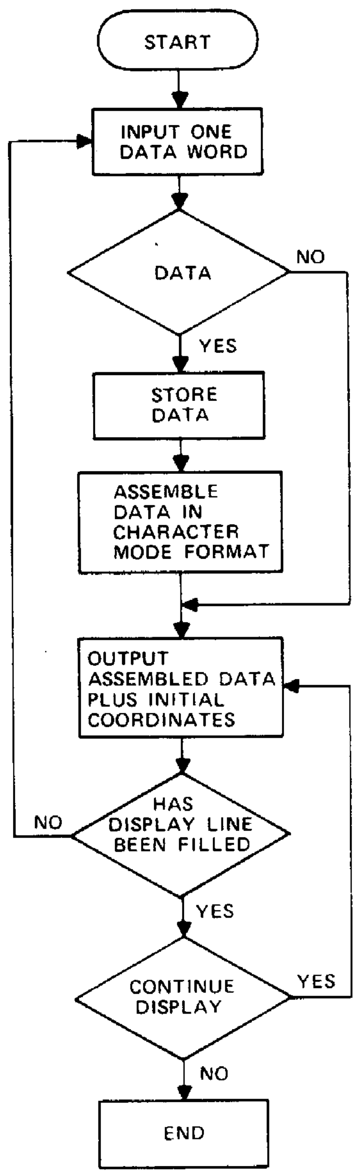


Figure 5-13. Receive and Display Program Flowchart

Real-Time Clock Programming

Channel 14_8 is reserved for the real-time clock. This channel, which is always active and full, and may be read at any time. The real-time clock is a 12-bit, free-running counter incrementing at a 1-MHertz rate from 0 through 4095_{10}.

Two-Port Multiplexer Programming

NOTE

For two-port multiplexer programming, bit numbering within words is 0 through 63 from left to right.

Channel 15_8 is reserved for communications with one or two external devices through the two-port multiplexer. One port is reserved for maintenance purposes, and the other is reserved for future use. The two-port multiplexer can communicate with all external devices that use EIA standard RS232C serial interface. The multiplexer can accommodate data with odd/even parity, 5 to 8 bits per character and 1 or 2 stop bits. Issuing appropriate function codes sets the format. The rate is switch selectable for each channel for operation between 110 and 9600 baud. These switches are located internally on the two-port multiplexer.

Two-Port Multiplexer Operation

The two-port multiplexer uses the rightmost 12 bits on channel 15_8. A 12-bit (octal) function word from the PP is translated to specify the following operating conditions.

Code	Function
7XXX	Terminal select.
6XXX	Terminal deselect.
00XX	Read status summary.
01XX	Read terminal data.
02XX	Write output buffer.
03XX	Set operation mode to terminal.
04XX	Set/clear terminal control signal, data terminal ready (DTR).
05XX	Set/clear terminal control signal, request to send (RTS).
06XX	Not used.
07XX	Master clear selected port.

Terminal Select (7XXX)

The PP sends this select code to specify the terminal to which the function codes and data transmissions apply. Code 7000 selects port 0 (for future use), and code 7001 selects port 1 (maintenance console).

Terminal Deselect (6XXX)

The PP sends this code, which deselects the two-port multiplexer from channel 15_8 so the 16-bit channel is available for inter-PP communications.

Read Status Summary (00XX)

This code permits the PP to input status from the selected terminal. One-word input must follow to read the status response. The response is 12 bits, which are defined as follows.

Bit	Status
52-58	Not used.
59	Output buffer not full.
60	Input ready.
61	Data carrier detect or carrier on.
62	Data set ready.
63	Ring indication.

PP Read Terminal Data (01XX)

This code permits the PP to input the terminal data from the selected terminal. Channel 15_8 must be activated, and a 1-word input must follow to read in the terminal data. The data word is 12 bits, which are defined as follows.

Bit	Status
52	Data set ready.
53	Data set ready and data carrier detector.
54	Over run.
55	Framing or parity error.
56-63	8-bit data.

Data Set Ready (Bit 52)

When the data set ready signal is active, this bit sets.

Data Set Ready (DSR) and Data Carrier Detector (DCD) (Bit 53)

When both data set ready and data carrier detector signals are active, this bit sets.

Over Run (Bit 54)

When the previously received character is not read by the PP before the present character is transferred to the data holding register, the overrun bit sets.

Framing or Parity Error (Bit 55)

When the received character does not have a valid stop bit (framing error) or when this bit sets, the received character parity does not agree with the select parity (parity error).

Data Character (Bits 56 Through 63)

The lower 8 bits of the input word form the data character. The multiplexer forms this character directly from the Universal Asynchronous Receiver and Transmitter (UART).

PP Write Output Buffer (02XX)

This code prepares the multiplexer for an output operation to the 64-character output buffer memory. Before an output operation can proceed, channel 15_8 must be activated. The output operation is terminated when the multiplexer receives an inactive signal from the PP or when no more locations are available in the output buffer. In the latter case, an inactive (instead of empty) signal is sent back to the channel, which in turn, terminates the output operations.

Set Operation Mode to the Terminal (03XX)

This code permits the PP to set the terminal operation mode register. A 12-bit function code word from the PP specifies the operation of the terminal. This word is decoded in the function register. Segments of the word define the mode as follows:

Bit	Status
58	Not used.
59	No parity. When this bit is set, it eliminates the parity bit from the transmitted and received characters. The stop bit(s) immediately follow the last data bit.
60	Number of stop bits. This bit selects the number of stop bits, 1 or 2, to be appended immediately after the parity bit. When this bit is clear, it inserts 1 stop bit and when set, it inserts 2 stop bits.
61-62	Number of bits per character. These 2 bits are internally decoded to select 5, 6, 7, or 8 data bits per character.

Bit 61	Bit 62	Bits Per Character
0	0	5
0	1	6
1	0	7
1	1	8

Bit	Status
63	Odd/even parity select. This bit selects the type of parity that will be appended immediately after the data bits. It also determines the parity that will be checked on read data.

Set/Clear Data Terminal Ready (04XX)

This code permits the PP to set or clear the terminal control signal, data terminal ready (DTR). When bit 63 is set, DTR is active, and when bit 63 is clear, DTR is inactive.

Set/Clear Request to Send (05XX)

This code permits the PP to set or clear the terminal control signal, request to send (RTS). When bit 63 is set, RTS is active, and when bit 63 is clear, RTS is inactive.

Master Clear (07XX)

This code permits the PP to master clear the selected port including its output buffer memory and UART. The terminal operation mode register and terminal control signals are not cleared.

Programming Considerations

Channel 15_8 communicates with the terminals connected to the external interface, one at a time. To establish communications between a PP and the terminal, the PP issues a function for select. The function word for select is formed by the least-significant 12 bits, which are sent to channel 15_8, and specifies the following information.

- A select code to select the multiplexer (7XXX).
- The terminal with which the PP would like to establish communication (7XXX).

When the connect is established, the two-port multiplexer routes all data to the terminal designated by the select code. The multiplexer responds with the inactive signal to acknowledge the receipt of the function code of 7XXX for select, 6XXX for deselect, and 0XXX for operation. Otherwise, the multiplexer ignores the function.

Output Data

The multiplexer accepts a maximum data block length of 64 characters per terminal. During the block data transfer, the multiplexer terminates the output operation either when it receives an inactive signal from the channel or when the output buffer is full. When the output buffer is full, the multiplexer sends back an inactive signal instead of an empty signal to the channel on the last output word. The signal indicates the output buffer accepts the last output word and it cannot receive anymore data from the PP. The multiplexer does not allow output to a full buffer. The multiplexer sends back an inactive signal to deactivate channel 15_8 after the multiplexer decodes the previous function code, which is 02XX (PP write output buffer), and receives an activate signal from the PP.

Input Data

The multiplexer does not store the input data from the terminal. A lost data condition exists if the PP does not input the previous data before the new data arrives from the terminal. The multiplexer allows input from an empty input buffer.

Request to Send and Data Terminal Ready

The hardware brings up request to send and data terminal ready automatically under the following conditions regardless of the software RTS and DTR bits.

- Data in the UART output register.
- Data in the FIFO output register.

When no data is in the FIFO or UART, the software bit determines RTS and DTR.

Maintenance Channel Programming

NOTE

Maintenance registers are numbered 0 through 63 from left to right.

Maintenance Channel

A PP in the IOU can perform any or all of the following operations through the maintenance channel (MCH) to each system element, such as the CP, IOU, and CM.

- Initializing registers, controls, and memories.
- Monitoring and recording error information.
- Verifying error-detection and correction hardware.

The maintenance channel consists of the maintenance channel interface on channel 17_8, a maintenance channel interface in each system element, and a set of interconnecting cables.

The IOU maintenance channel interface contains a selector that connects to one of up to seven system elements. The IOU is element 0, and its maintenance access control is internally connected to the selector. All other system elements are assigned arbitrary element numbers. A single cable connects each maintenance access control to the selector. This arrangement results in a radial connection that allows any system element to be shut down or removed without affecting communication with the other elements.

MCH Function Words

The MCH function word consists of the connect, opcode, and type fields, which are used as described in the next three paragraphs and tables 5-11 and 5-12.

The connect field specifies the unit to which the MCH is connected (CP, CM, or IOU), controlling selection within the IOU only. The unit remains connected until another connect code selects a different unit. Connect codes 10_8 to 17_8 leave the MCH unconnected; in this state, the interface can be used for PP to PP communications.

The OPCODE field controls the unit selected by the connect code, preparing the unit for a coming read/write/echo operation or causing the unit to halt, start, clear, or deadstart.

The use of the TYPE field depends on the connected unit. When the CP is the connected unit, type codes 1 through 7 specify the data type in the operation to be performed. Also, for the CP, type code 0 specifies that the internal address of the CP register to be connected is specified in a control word, which is sent as 2 data words immediately following the function word. When IOU is the connected unit, type codes 0 through 7 specify the starting byte number for read/write operations. The exceptions are reading the options installed and element identifier registers. CM uses A_{16} to access the maintenance registers.

Table 5-11. Bit Assignments for MCH Function Word to CP and CM

Field			MCH Function Word to CP and CM
TYPE (bits 0-3)	Code 0_{16}	=	CP and CP registers.
OPCODE (bits 4-7)	Code 0_{16}	=	Halt processor.
	1_{16}	=	Start processor.
	4_{16}	=	Prepare for read.
	5_{16}	=	Prepare for write.
	6_{16}	=	Master clear.
	7_{16}	=	Clear errors.
TYPE (bits 0-3)	Code 1_{16}	=	Control store memory.
OPCODE (bits 4-7)	Code 4_{16}	=	Prepare for read.
	5_{16}	=	Prepare for write.
TYPE (bits 0-3)	Code $3\text{-}7_{16}$	=	Internal memories.
OPCODE (bits 4-7)	Code 4_{16}	=	Prepare for read.
	5_{16}	=	Prepare to write.
TYPE (bits 0-3)	Code A_{16}	=	CM and CM registers.
OPCODE (bits 4-7)	Code 4_{16}	=	Prepare for read.
	5_{16}	=	Prepare for write.
	6_{16}	=	Master clear.
	7_{16}	=	Clear errors.

Table 5-12. Bit Assignments for MCH Function Word to IOU

Field			MCH Function Word to IOU
CONNECT (bits 8-11)	Code 0_{16}	=	Connect IOU maintenance registers.
OPCODE (bits 4-7)	Code 4_{16}	=	Prepare for read (control word required).
	5_{16}	=	Prepare for write (control word required).
	6_{16}	=	Master clear.
	7_{16}	=	Clear fault status registers.
	C_{16}	=	Read IOU status summary (reads 1 byte, control word not required).
TYPE (bits 0-3)	Codes $0\text{-}7_{16}$	=	IOU registers are read circularly (byte 0 follows byte 7) from the byte specified by the TYPE field.

MCH Control Words

Some function words must be followed by two 8-bit control words, which specify the internal address of the register to be accessed. This is accomplished by transmitting two PP words where the rightmost 8 bits in each word are used. Control words are required for the following.

- Function words to CP with opcodes 4/5.
- Function words to CM and IOU with opcodes 4/5.
- Function words to CP, CM, and IOU with opcode 8 (echo).

Refer to tables 5-13 through 5-15 for CP, CM, and IOU internal address assignments.

MCH Programming for Halt/Start (Opcode 0/1)

These operations consist of the output of a function word. A halt opcode halts the processor without damaging the process being executed, including the integrity of the interunit communication of the halted processor such as CDC CYBER 170 exchange request communication, central memory communications, and the process state. If the process is subsequently restarted without performing any other MCH operations or after performing read/write with certain precautions, the process continues without damage.

MCH Clear LED (Opcode 3)

This operation clears all LEDs associated with pak errors and is intended, but not required, for use at system initialization. For maintenance reasons, this operation can also clear LEDs without initializing and master-clearing.

MCH Programming for Read/Write (Opcode 4/5)

Refer to Programming for PP Data Input/Output in this chapter for a more complete procedure. In general terms, proceed as follows:

1. Issue the function with opcode 4/5.
2. Output the first control word.
3. Verify the error flag is clear.
4. Output the second control word.
5. Verify the error flag is clear.
6. Input/output the required number of data words.
7. Verify the error flag is clear.

Reading a nonexistent register returns all 0's. Writing to a read-only register or to a nonexistent register does not alter any register. Most registers are read/write as 64-bit (8-byte) registers, requiring the input/output of 8 MCH data words. Most registers that are physically smaller than 8 bytes are right-justified with zero-fill. Exceptions are as follows:

- Reading a status summary register repeats the status information in each byte.
- The IOU may disconnect the MCH without affecting subsequent MCH operations in the following cases:
  - After reading 1 to 8 bytes from any maintenance register.
  - After writing 1 byte to a corrected error log register.
  - After writing 1 byte to an uncorrected error log register.

The following MCH operations on CP registers can be performed with the CP running or halted.

- Read CP status summary register.
- Read CP fault status register.
- Read CP corrected error log registers.
- Read CP options installed registers.
- Read CP element identifier register.
- Read/write CP dependent environmental control register.
- Read/write test mode control registers.
- Clear errors.

To read/write other CP-registers, the CP must be running since these registers are accessed by microcode. Refer to the Maintenance Register Codes Booklet listed in the preface for register bit assignments.

MCH Programming for Master Clear/Clear Errors (Opcode 6/7)

These operations consist of the output of a single function word. The master clear immediately and arbitrarily clears the connected unit without regard to possible information loss. Clear errors clears the error indicators in the connected unit. To avoid loss of error information while the errors are cleared, the unit concerned should be halted.

MCH Echo (Opcode 8)

This operation checks the data path between the MCH and the IOU MAC. Following the operation MCH is activated and 2 bytes are sent to IOU MAC. IOU ignores the first byte and latches the second byte in the Address Holding Register in any data pattern. MCH is deactivated after the second byte is accepted in IOU MAC, and the channel is activated followed by an input sequence. IOU MAC sends data (contents of Address Holding Register) upon receiving the Active signal and subsequent Empty signals. There is no restriction on the number of data words read.

MCH Programming for Read IOU Status Summary (Opcode C, IOU Only)

This operation is an alternative, faster means of reading the IOU status summary register.

1. Issue function with opcode C.

2. Input status summary byte.

Table 5-13. CP Internal Address Assignments

Internal Address (1) Hex	Octal	Type (2)	Type (3)	Description
00	000	R	A	Status summary register.
10	020	R	A	Element identifier register.
30	060	R	A	Dependent environment control register.
42	082	R	M	Monitor condition register.
80-89	200-211	R	A	Processor fault status registers 1 through 9.

Notes:

(1) The internal address is the second byte of two 8-bit control words, which must be supplied after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write.

(3) A = always accessible, M = microcode accessible.

Table 5-14. CM Internal Address Assignments

Internal Address (1) Hex	Octal	Type (2)	Description
00	000	R	Status summary register.
10	020	R	Element identifier register.
12	022	R	Options installed register.
A0	240	R/W	Corrected error log register.
A4	244	R/W	Uncorrected error log 1 register.
A8	250	R/W	Uncorrected error log 2 register.

Notes:

(1) The internal address is the second byte of two 8-bit control words, which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write.

Table 5-15. IOU Internal Address Assignments

Internal Address (1)			
Hex	Octal	Type (2)	Description
00	000	R	Status summary register.
10	020	R	Element identifier register.
12	022	R	Options installed register.
18	030	R/W	Fault status mask register.
40	100	R	Status register.
80	200	R/W	Fault status 1 register.
81	201	R/W	Fault status 2 register.
A0	240	R/W	Test mode.

Notes:

(1) The internal address is the second byte of two 8-bit control words, which must be issued after a function word output with OPCODE = 4/5. The first byte is discarded.

(2) R = read, W = write.

Appendix

Glossary

A

A

ADU	Assembly-disassembly unit
AOR	Address out of range

C

CEL	Corrected error log
CIF	CMU interrupted flag
CIO	Concurrent input/output
CM	Central memory
CMU	Compare/move unit
CP	Central processor
CRT	Cathode-ray tube
CTI	Common Test and Initialization

D

DMA	Direct-memory access
DSC	Display station
DTR	Data terminal ready

E

ECC	Error correction code
ECL	Emitter-coupled logic
EDS	Extended deadstart
EIA	Electronic Industries Association
EM, EMS	Exit mode selection
EC	Exit condition code field at (RAC)

F

FIFO	First in, first out
FLC	Field length, central memory
FLE	Field length, extended memory

H

HIVS	Hardware Initialization and Verification Software

I

ILH	Instruction lookahead hardware
I/O	Input/output
IOU	Input/output unit
IPI	Intelligent peripheral interface
ISI	Intelligent standard interface

M

MA	Monitor address
MCH	Maintenance channel
MF	Monitor flag
MOS	Metal oxide semiconductor
MUX	Multiplexer, selector

N

NIO	Nonconcurrent input/output

O

OS	Operating system

P

PE	Parity error
PP	Peripheral processor
PPM	Peripheral processor memory

R

RAC	Reference address, central memory
RAE	Reference address, extended memory
RAM	Random access (read-write) memory
RNI	Read next instruction
ROM	Read-only memory
RTS	Request to send

S

SECDED	Single-error correction double-error detection

U

UART	Universal Asynchronous Receiver and Transmitter
UEM	Unified extended memory

Index

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Z

COMMENT SHEET

MANUAL TITLE: CYBER 840A, 850A, 860A, 870A Computer Systems
CYBER 170 State Hardware Reference Manual

PUBLICATION NO.: 60463560 REVISION: C

NAME: ____________________

COMPANY: ____________________

STREET ADDRESS: ____________________

CITY: ____________ STATE: ____________ ZIP CODE: ____________

This form is not intended to be used as an order blank. Control Data Corporation welcomes your evaluation of this manual. Please indicate any errors, suggested additions or deletions, or general comments below (please include page number references).

☐ Please Reply ☐ No Reply Necessary

CUT ALONG LINE

REV. 5/86 PRINTED IN U.S.A.

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

FOLD ON DOTTED LINES AND TAPE

FOLD

FOLD

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO 8241 MINNEAPOLIS, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CONTROL DATA

Technology and Publications Division
ARH219
4201 North Lexington Avenue
Saint Paul, MN 55126-6198

FOLD

FOLD

CUT ALONG LINE