CRAY X-MP EA Computer Systems Functional Description Manual

HR-3020

Cray Research, Inc.

Requests for copies of Cray Research, Inc. publications should be directed to:

CRAY RESEARCH, INC.
Distribution Center
2360 Pilot Knob Road
Mendota Heights, MN 55120

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
HARDWARE PUBLICATIONS
890 Industrial Blvd.
Technical Operations Building
Chippewa Falls, WI 54729

Record of Revision

Each time this manual is revised and reprinted, all changes issued against the previous version are incorporated into the new version and the new version is assigned an alphabetic level which is indicated in the publication number on each page of the manual.

Changes to part of a page are indicated by a change bar in the margin directly opposite the change. A change bar in the footer indicates that most, if not all, of the page is new. If the manual is rewritten, the revision level changes but the manual does not contain change bars.

REVISION	DESCRIPTION
	April, 1988 - Original printing.

PREFACE

This manual describes the basic functions of the CRAY X-MP Extended Architecture (EA) computer system manufactured by Cray Research, Inc. (CRI).

AUDIENCE

This manual is written primarily for customers. It describes the design and architecture of the CRAY X-MP EA computer system and its associated peripheral devices.

ORGANIZATION

This manual is organized into the following tabbed sections. A detailed table of contents is included at the beginning of each tabbed section.

SECTION 1 - CRAY X-MP EA COMPUTER SYSTEM OVERVIEW - This section introduces and describes the CRAY X-MP EA system components and support equipment.

SECTION 2 - CRAY X-MP EA MAINFRAME - This section describes the basic architecture of the CRAY X-MP EA mainframe and is divided into two subsections. The first subsection describes the hardware architecture of the mainframe. The second subsection describes the CPU instructions. Specification sheets for the CRAY X-MP EA computer system are included at the end of this section.

SECTION 3 - I/O SUBSYSTEM - This section describes the basic architecture and functions of the I/O Subsystem (IOS). A specification sheet for the IOS is included at the end of this section.

SECTION 4 - SSD SOLID-STATE STORAGE DEVICE - This section describes the basic architecture and functions of the SSD solid-state storage device. A specification sheet for the SSD is included at the end of this section.

SECTION 5 - PERIPHERAL EQUIPMENT - This section describes the function of the disk drives and network interface equipment used by the CRAY X-MP EA computer system. Specification sheets for the the different disk drives and network interfaces are included at the end of this section.

SECTION 6 - SOFTWARE OVERVIEW - This section provides an overview of the software available for the CRAY X-MP EA computer system.

For the reader's convenience, a glossary is included. It defines many of the commonly used abbreviations and terminology associated with Cray computer systems.

NOTATIONAL CONVENTIONS

The following conventions are used throughout this manual.

Convention	Description
Lowercase italic	Variable information.
X or x or *x*	An unused variable.
n	A specified value.
(value)	The contents of the register or memory location designated by value.
Register bit designators	Register bits are numbered from right to left as powers of 2. Bit 2^0 corresponds to the least significant bit of the register, With the exception of the Vector Mask register. The Vector Mask register bits correspond to a word element in a vector register; bit 2^{63} corresponds to element 0 and bit 2^0 corresponds to element 63.
Number base	All numbers used in this manual are decimal, unless otherwise indicated. Octal numbers are indicated with an 8 subscript. Exceptions are register numbers, the instruction parcel in instruction buffers, and instruction forms, which are given in octal without the subscript.

The following are examples of the preceding conventions.

Example	Description
Transmit (A*k*) to S*i*	Transmit the contents of the A register specified by the *k* field to the S register specified by the *i* field.
167*ixk*	Machine instruction 167. The *j* field is not used.
Read n words from memory	Read a specified number of words from memory.
Bit 2^{63}	The value represents the most significant bit of an S register or element of a V register.
1000_8	The number base is octal.

RELATED PUBLICATIONS

For additional information on site planning refer to the following publications:

• HR-0080	*Cray Peripheral Equipment Site Planning Reference Manual.* This manual provides site planning information for operator and maintenance workstation equipment, Disk Storage Units (DSUs), and Front-end Interface (FEI) cabinets.
• HR-0082	*Cray Support Equipment Site Planning Reference Manual.* This manual provides site planning information for refrigeration condensing units (RCUs) and motor-generator sets (MGS).
• HR-3017	*CRAY X-MP EA 6-column Computer Systems Site Planning Reference Manual.* This manual provides site planning information for the CRAY X-MP EA 6-column mainframe, the I/O Subsystem (IOS), and the SSD solid-state storage device.
• HR-3018	*CRAY X-MP EA 8-column Computer Systems Site Planning Reference Manual.* This manual provides site planning information for the CRAY X-MP EA 8-column mainframe, the I/O Subsystem (IOS), and the SSD solid-state storage device.
• HR-3019	*CRAY X-MP EA 12-column Computer Systems Site Planning Reference Manual.* This manual provides site planning information for the CRAY X-MP EA 12-column mainframe, the I/O Subsystem (IOS), and the SSD solid-state storage device.

A list of related software publications is included at the end of Section 6.

CONTENTS

CONTENTS

1 - CRAY X-MP EA COMPUTER SYSTEM OVERVIEW

The CRAY X-MP EA computer system is a powerful general-purpose supercomputer that contains one or more central processing units (CPUs). The CRAY X-MP EA computer system is able to achieve extremely rapid multiple processor rates by efficiently using the scalar and vector processing capabilities of the CPUs, combined with the systems solid-state, random access memory (RAM) and shared registers. The CRAY X-MP EA computer system is carefully balanced to deliver optimum overall performance. The architecture of CRAY X-MP EA computer systems allows faster and more efficient use of the vector and scalar processing capabilities inherent in all Cray computer systems.

Vector processing uses a single instruction to perform multiple operations on sets of ordered data; whereas scalar processing is a sequential operation during which one instruction produces one result. When two or more vector operations are chained together, two or more operations run simultaneously. Therefore, the computing rate for vector processing greatly exceeds that of conventional scalar processing. Scalar operations complement the vector operations by providing solutions to problems not readily adaptable to vector techniques.

The start-up time for vector operations on the CRAY X-MP EA computer system is short, therefore vector processing is more efficient than scalar processing for vectors containing as few as two elements. This feature allows for fast long and short vector processing to be balanced with high-speed scalar processing, while both are supported by powerful input/output capabilities.

The multiple processors of the CRAY X-MP EA computer system allows the use of multiple processing or multitasking techniques. Multiple processing allows several programs to run concurrently on multiple CPUs of a single mainframe. Multitasking allows two or more parts of a program to run in parallel, sharing a common memory space.

The maximum configuration of the CRAY X-MP EA/4 Computer System is shown in Figure 1-1. Mass storage devices (such as disk and tape drives) and front-end interfaces (FEIs) can also be configured with the system.

Support equipment for the CRAY X-MP EA computer system includes refrigeration condensing units (RCUs) for refrigeration, motor generators sets (MGS) for system power, and power distribution units (PDUs) for proper power distribution to the mainframe, IOS, and SSD.

The following subsections introduce the computer system components; later sections provide more detailed information on the IOS, FEIs, SSD, and mass storage devices.

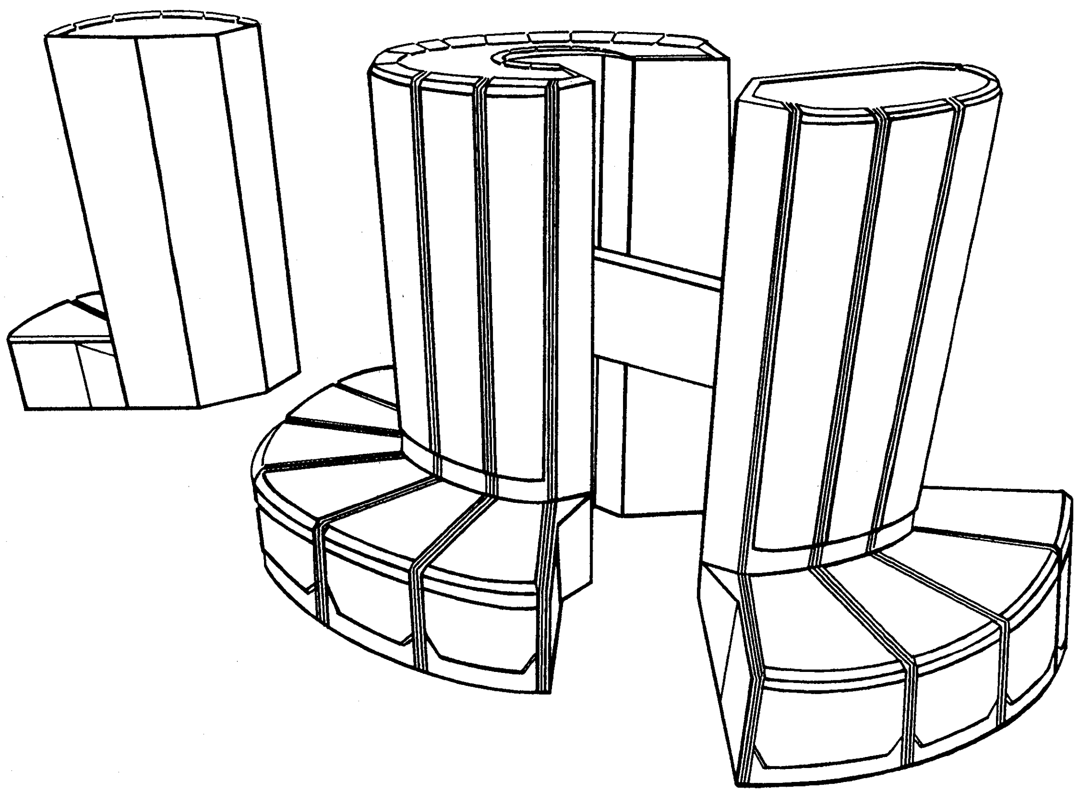


Figure 1-1. Maximum Configuration of a CRAY X-MP EA/4 Computer System

CRAY X-MP EA MAINFRAME

The CRAY X-MP EA mainframe contains one or more central processing units (CPUs), an I/O section, a Real-time Clock (RTC), and Central Memory. In multiple-processor mainframes, the I/O section, the Interprocessor Communication section (which controls processing between individual CPUs and between CPUs Central Memory), the Real-time Clock, and Central Memory are shared by the CPUs.

Each CPU has a computation section consisting of operating registers, functional units, and a control section. The control section determines instruction issue and coordinates the three types of processing (vector, scalar, or address).

Refer to Section 2 for more specific information on the CRAY X-MP EA mainframe.

I/O SUBSYSTEM

The CRAY X-MP EA computer system includes an I/O Subsystem (IOS). The IOS chassis shown in Figure 1-2 has multiple I/O Processors (IOPs), a Buffer Memory, and required interfaces. It is designed for rapid data transfer between front-end computers, peripheral devices, storage devices, and the IOS's Buffer Memory, or between its Buffer Memory and the Central Memory of the CRAY X-MP EA mainframe.

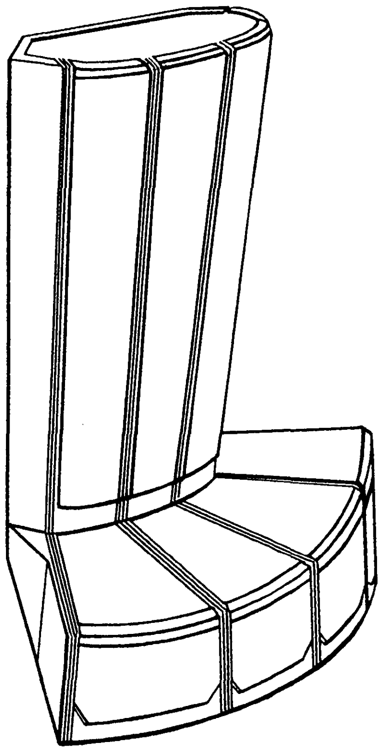


Figure 1-2. I/O Subsystem Chassis

The IOS is configured with a variety of different IOPs; each IOP controls different portions of the CRAY X-MP EA computer system. The number of IOPs configured with a system is site-dependent. Each IOP has a memory section, a control section, a computation section, and an I/O section. I/O sections are independent and handle some portion of the I/O requirements for the IOS. IOS hardware allows simultaneous data transfers between the IOPs and the mainframe's Central Memory over 100-Mbyte/s I/O channels.

The IOS also interfaces with the High-speed External (HSX) communications channel. The HSX channel connects peripheral equipment such as high-speed graphic devices to the CRAY X-MP EA mainframe. Cray Research, Inc. (CRI) does not provide the external peripheral equipment but does provide the hardware connections and software drivers for the channel.

The HSX channel can also be connected to the SSD. With this configuration, data moves between Central Memory and the SSD over the conventional SSD channel and then transfers to the IOS/HSX channel.

Refer to Section 3 of this manual for more specific information on the IOS.

SSD SOLID-STATE STORAGE DEVICE

The SSD is an optional high-performance device used for temporary data storage. Figure 1-3 shows an SSD chassis. The SSD transfers data between the mainframe's Central Memory and the SSD through one or two special 1,000-Mbyte/s channels. The speed of these transfers depends on the SSD and CRAY X-MP EA system configuration. The SSD can also be connected directly to an IOS over a 100-Mbyte/s channel pair. The SSD-3I and SSD-5I are special versions of the SSD that are housed within the IOS chassis.

Refer to Section 4 of this manual for specific information on the SSD.

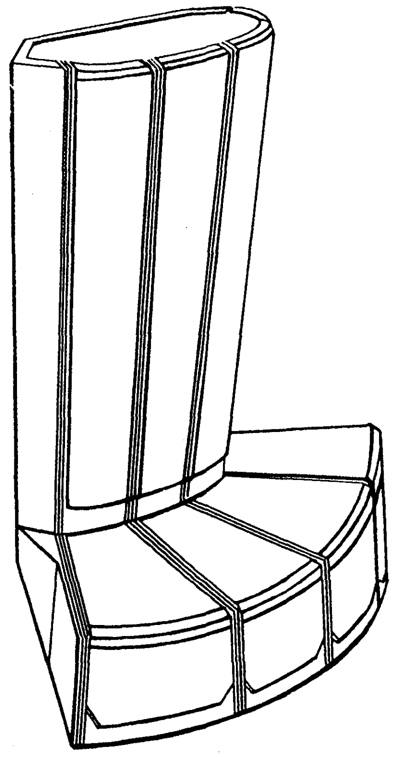


Figure 1-3. SSD Solid-state Storage Device Chassis

DISK STORAGE UNITS

For mass storage, the CRAY X-MP EA computer system uses CRI Disk Storage Units (DSUs). A Disk Controller Unit (DCU) interfaces the DSUs to an IOP within the IOS. The IOP and the DCU can transfer data between the IOP and multiple DSUs without missing data or skipping revolutions even when all DSUs are operating at full speed. Refer to Section 5 of this manual for more specific information on Disk Storage Units.

COMMUNICATION INTERFACES

The CRAY X-MP EA mainframe is designed to communicate easily with front-end computer systems and computer networks.

Standard FEIs connect either the I/O channels of the CRAY X-MP EA mainframe or the IOS to channels of front-end computers. This connection provides input data to the CRAY X-MP EA computer system and receives output from it for distribution to peripheral equipment. An FEI compensates for differences in channel widths, machine word size, electrical logic levels, and control signals.

Some FEIs are housed in stand-alone cabinets located near the host computer (refer to Figure 1-4), while some are installed directly into the front-end computer system. In either case, operation of the FEI is invisible to the Cray user.

An optional fiber-optic link is available for FEIs that provides front-end connections of up to 3,280 ft (1,000 m) separation from the CRAY X-MP EA computer system.

The CRAY X-MP EA mainframe can be connected to computer networks directly or through a front-end computer system. Refer to Section 5 of this manual for specific information on Communication Interfaces.

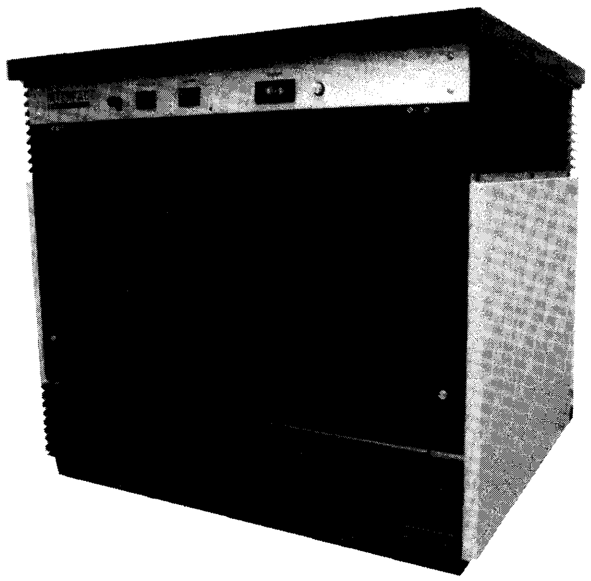


Figure 1-4. Typical Front-end Interface Cabinet

OPERATOR AND MAINTENANCE WORKSTATIONS

VMEbus technology is used to provide two workstations on the CRAY X-MP EA computer system: the Operator Workstation (OWS) and the Maintenance Workstation (MWS). Both workstations run UNIX System V software. The OWS is a microcomputer system that provides the following:

- System operator interface
- System deadstart and master clear functions
- Software maintenance utilities
- Local tape and local printer control
- Time-of-day clock
- Remote access for software and hardware service
- System monitoring activities requiring security

In addition, the OWS provides enhanced features such as the Ethernet interface, which can be used to link workstations in a network for multiple-system operators or sites.

The OWS communicates with the CRAY X-MP EA computer system through a 6-Mbyte/s I/O channel from an IOP in the IOS. The tape drives, disks, printer, and time-of-day clock are under direct control of the OWS.

The MWS is microcomputer system used for hardware maintenance and monitoring. The MWS provides the following:

- Off-line diagnostics for testing the mainframe, IOS, SSD, and peripherals
- System deadstart and master clear functions
- System error logging

The MWS communicates with the CRAY X-MP EA computer system through a 6-Mbyte/s channel from an IOP located in the IOS. The MWS monitors a special system error channel to detect and log system errors, such as double-bit memory errors.

POWER AND COOLING SUPPORT EQUIPMENT

CRAY X-MP EA computer systems require support equipment for power and cooling. Power is supplied by a Motor Generator Set (MGS) and Power Distribution Unit (PDU). The remainder of this section defines and explains the various mainframe, IOS, and SSD support equipment. Refer to the appropriate Site Planning manuals listed in the Preface for more information on power and cooling requirements.

The CRAY X-MP EA mainframe power supplies and voltage-adjusting controls are located in the mainframe's PDU. The 400-Hz power from the MGS is distributed among the power supplies (motor generators are described later in this section).

To control the intense heat created by the density of the circuitry, the CRAY X-MP EA mainframe has a monitoring system consisting of PC boards and a system control panel. The monitoring system provides a method for the system attendant to observe these conditions. The monitoring system also functions as a backup system that can automatically shut down the mainframe if an abnormal condition develops and the system attendant fails to notice the condition. In addition to these automatic monitoring features, the AC output voltage on the MGS can be set from this panel.

The refrigeration condensing unit (RCU) contains the major components of the refrigeration system used to cool the mainframe, IOS, and SSD. Heat is transferred from the RCU by customer-supplied cooling water. Figure 1-5 shows the RCU for a CRAY X-MP EA 1, 2, or 4 processor mainframe only; the CRAY X-MP EA/se uses a smaller RCU. The number of RCUs needed is site dependent.

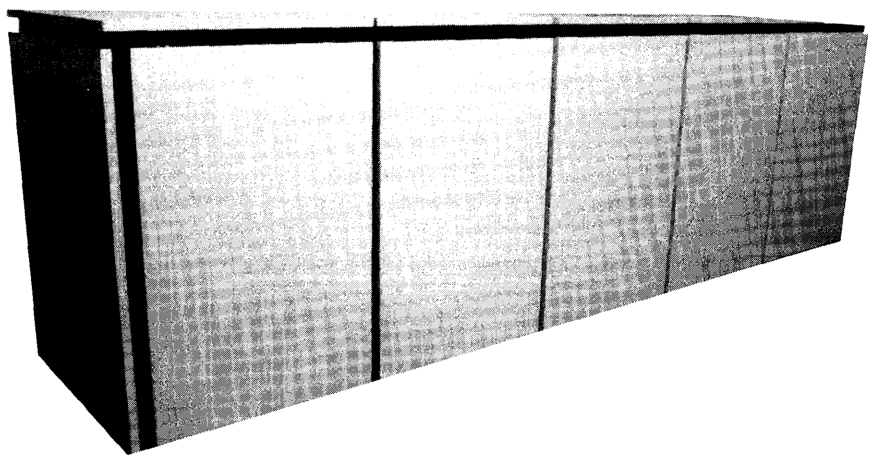


Figure 1-5. Refrigeration Condensing Unit

Each Motor Generator (MG) converts primary power from commercial power mains to the 400-Hz power used by the mainframe, IOS, and SSD. The MG isolates these components from transients and fluctuations on the commercial power mains. A MG cabinet is shown in Figure 1-6.

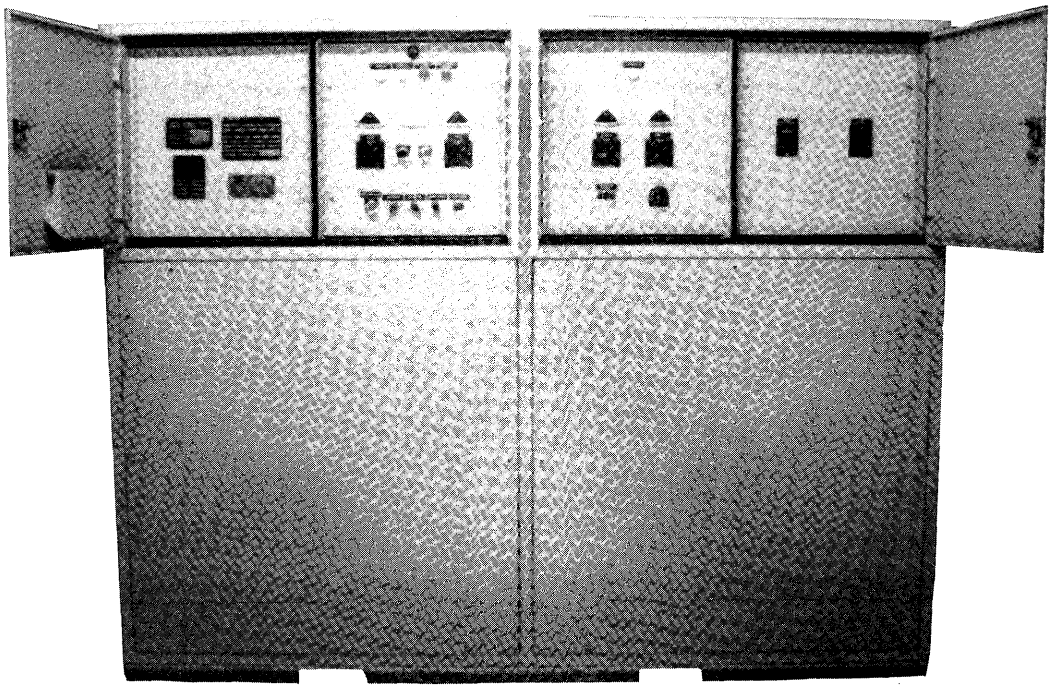


Figure 1-6. Motor Generator Cabinet

The mainframe, IOS, and SSD each have independent PDUs. The PDU for the mainframe contains adjustable transformers for regulating the voltage to each column of the mainframe. The PDU also contains temperature and voltage monitoring equipment that checks temperatures at strategic locations on the mainframe chassis (the automatic warning system alerts the system attendant if an overheating or excessive cooling occurs). If the system attendant fails to notice these conditions, the PDU powers down the equipment. Control switches for the MGS and the RCU are also mounted on the mainframe's PDU.

Figure 1-7 shows the PDUs for the CRAY X-MP EA/4 processor mainframe (left) and for the CRAY X-MP EA/1 and 2 processor mainframes, IOS, and SSD (right).

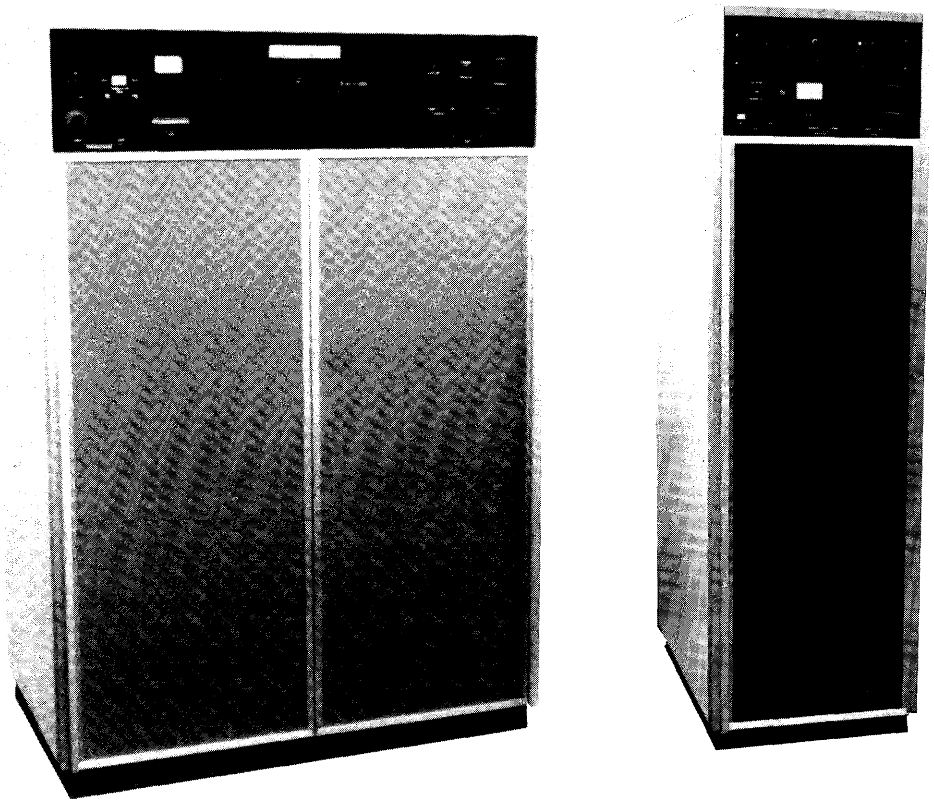


Figure 1-7. Power Distribution Units

CONTENTS

FIGURES

FIGURES (continued)

TABLES

2 - CRAY X-MP EA MAINFRAME

This section describes the major functional areas and special features of a CRAY X-MP EA mainframe, and provides a summary of the Cray Assembly Language (CAL) instruction set.

CPU SHARED RESOURCES

The central processing units (CPUs) of the multiple processor CRAY X-MP EA computer systems share several functional areas (or sections) of the mainframe. These sections are Central Memory, the I/O section, the Interprocessor Communication section, and the Real-time Clock. The following subsections describe these functional areas.

Central Memory

For multiple processor CRAY X-MP EA computer systems, Central Memory is shared by the CPUs and the I/O section. Central Memory is divided into interleaved banks. This arrangement improves memory-access speed by allowing simultaneous and overlapping memory references. Refer to the specification mainframe sheets at the end of this section for more information on memory size and number of banks.

Each CPU in the system has four parallel memory ports. Each port performs specific functions, allowing different types of memory transfers to occur simultaneously. To further enhance memory operations, the bidirectional memory mode allows block read and write operations to occur concurrently.

The CRAY X-MP EA computer system has built-in resolution hardware to minimize the delays caused by memory conflicts and to maintain the integrity of all memory references when conflicts occur. A memory conflict occurs when more than one reference is made to the same area of Central Memory.

To protect data, single-error correction/double-error detection (SECDED) logic is used in Central Memory and on data channels to or from Central Memory. When data is written into Central Memory, a checkbyte (an 8-bit Hamming † code) is generated for the word and stored with that word. When the word is read from Central Memory, the checkbyte and data word are processed to determine if any bits are altered. If no errors occur, the word is passed without modification.

† Hamming, R. W., "Error Detection and Correcting Codes," *Bell System Technical Journal, 29, No. 2* (April 1950): pages 147-160.

If an error occurs, the 8 bits of the checkbyte are analyzed by the logic to find the number of altered bits. If only a single bit is altered, the correction logic resets that bit to the correct state and passes the corrected word on. The Memory Error flag in the Exchange Package sets to indicate that an error occurred, which can generate an interrupt. (Refer to "Flag Register Field" in this section for more information on the Memory Error flag.) Error information is also sent to an error logger.

If more than a single bit is altered, the logic cannot correct the word and the results are unpredictable. When a double error is detected, the Memory Error flag in the Exchange Package sets to indicate an error occurred, which can generate an interrupt signal. Error information is also sent to an error logger.

I/O Section

The I/O section is shared by the CPUs and may be equipped with a variety of high-performance channels for communication between the mainframe, the I/O Subsystem (IOS), and the SSD solid-state storage device. The latter two devices are high-speed data transfer devices designed to increase CRAY X-MP EA mainframe processing speeds. Refer to the mainframe specification sheets at the end of this section for more information on channel types and transfer rates.

Interprocessor Communication Section

The interprocessor communication section of the mainframe contains clusters of shared registers for interprocessor communication and synchronization. Each cluster consists of Shared Address (SB), Shared Scalar (ST), and Semaphore (SM) registers.

The SB and ST registers pass address and scalar information from one CPU to another, while the SM registers control activity between CPUs.

Each CPU Cluster Number (CLN) register determines which cluster of shared registers is accessed by a CPU. The cluster may be accessed by any CPU to which it is allocated in either user or system (monitor) mode. Any CPU in monitor mode can interrupt any other CPU and cause it to switch from user to monitor mode. The hardware also provides built-in detection of system deadlock within the cluster; a deadlock condition occurs when all CPUs in a cluster are holding issue on a Test and Set instruction.

Real-time Clock

The mainframe contains one Real-time Clock (RTC) that is shared by all the CPUs. This clock consists of a 64-bit counter that advances one count each clock period (CP). Because the clock advances synchronously with program execution, it can be used to time the program to an exact number of CPs. Contents of the RTC register can be read into or loaded from a Scalar (S) register.

CPU COMPUTATION SECTION

Each CPU is an identical, independent computing section consisting of operating registers, functional units, and an instruction control network (refer to Figure 2-1). The operating registers and functional units of each computing section are associated with three types of processing: address, scalar, and vector.

Address processing operates on internal control information such as loop counts, addresses, and indexes. This processing is done by Address (A) registers and dedicated integer arithmetic functional units.

Address information flows from Central Memory, instruction values, or control registers to A registers. Information in the A registers is distributed to various parts of the control network for use in controlling the scalar, vector, and I/O operations. The A registers can also supply operands to two integer functional units. The units generate address and index information and return the result to the A registers. Address information can also be transmitted to Central Memory from the A registers.

Scalar and vector processing are performed on data; scalar processing occurs sequentially and uses one operand or operand pair to produce a single result. Scalar processing is performed using Scalar (S) registers, several functional units dedicated solely to scalar processing, and additional floating-point functional units shared with vector operations.

Vector processing allows a single operation to be performed concurrently on a set (or vector) of operands, repeating the same function to produce a series of results. Vector processing is performed by Vector (V) registers, several functional units dedicated solely to vector processing, and additional floating-point functional units supporting both scalar and vector operations.

The main advantage of vector processing over scalar processing is that it eliminates instruction start-up time for all but the first operand. Start-up time for vector operations is short, therefore vector processing is more efficient than scalar processing for vectors containing as few as two elements. Register-to-register vector instructions eliminate the problem of memory conflicts.

Data flow in a computing section is from Central Memory to registers and from registers to functional units. Results flow from functional units to registers and from registers to Central Memory or back to functional units. Data flows along either the scalar or vector path, depending on the processing mode. An exception is that scalar registers can provide one required operand for some vector instructions.

The computation section performs integer or floating-point arithmetic operations. Integer arithmetic is performed in two's complement mode. Floating-point quantities have signed magnitude representation.

Integer or fixed-point operations are integer addition, integer subtraction, and integer multiplication. No integer division instruction is provided; the operation is accomplished through a software algorithm using floating-point hardware.

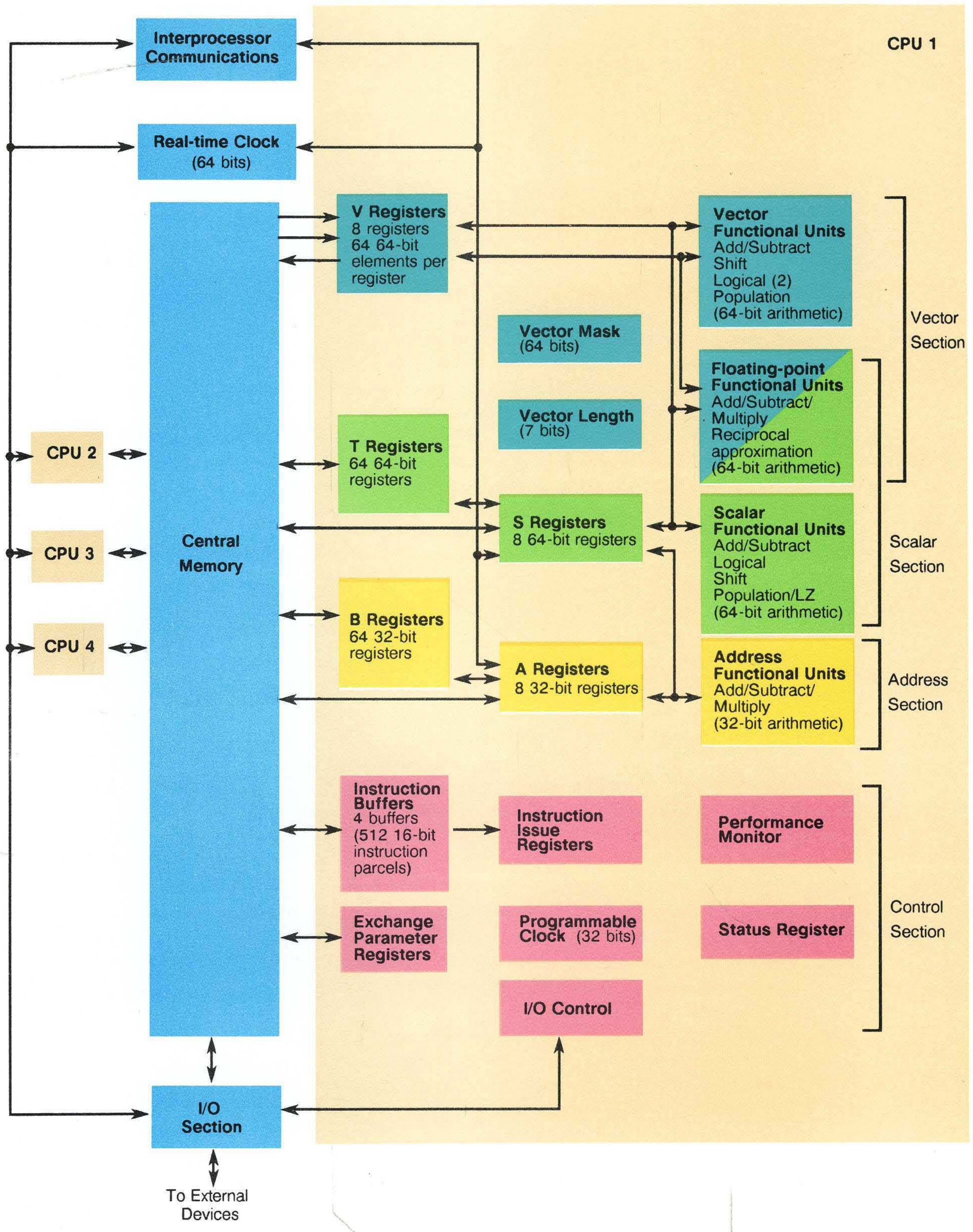

CPU 1
Interprocessor Communications
Real-time Clock (64 bits)
Central Memory
CPU 2
CPU 3
CPU 4
V Registers 8 registers 64 64-bit elements per register
T Registers 64 64-bit registers
B Registers 64 32-bit registers
Vector Mask (64 bits)
Vector Length (7 bits)
S Registers 8 64-bit registers
A Registers 8 32-bit registers
Vector Functional Units Add/Subtract Shift Logical (2) Population (64-bit arithmetic)
Floating-point Functional Units Add/Subtract/ Multiply Reciprocal approximation (64-bit arithmetic)
Scalar Functional Units Add/Subtract Logical Shift Population/LZ (64-bit arithmetic)
Address Functional Units Add/Subtract/ Multiply (32-bit arithmetic)
Vector Section
Scalar Section
Address Section
Instruction Buffers 4 buffers (512 16-bit instruction parcels)
Instruction Issue Registers
Performance Monitor
Exchange Parameter Registers
Programmable Clock (32 bits)
Status Register
I/O Control
Control Section
I/O Section
To External Devices

Figure 2-1. Multiple-CPU CRAY X-MP EA Computer System

Floating-point instructions include addition, subtraction, multiplication, and reciprocal approximation. The reciprocal approximation instructions can be used with a multiplication instruction sequence to perform a floating-point division operation.

The instruction set includes logical operations for AND, inclusive OR, exclusive OR, exclusive NOR, and a mask-controlled merge operation. Shift operations allow the manipulation of either 64-bit or 128-bit operands to produce 64-bit results. With the exception of address integer arithmetic, most operations are implemented in vector and scalar instructions.

The integer product is a scalar instruction designed for index calculation. Full indexing capability is possible throughout memory in either scalar or vector mode. The index can be positive or negative in either mode. Indexing allows matrix operations in vector mode to be performed on rows or on the diagonal as well as allowing conventional column-oriented operations.

Population and parity count instructions are provided for both vector and scalar operations. An additional scalar operation is the leading zero count.

Registers

Each CPU has three primary and two intermediate sets of registers. The primary sets of registers are the Address (A), Scalar (S), and Vector (V) registers. These registers are considered primary because Central Memory and the functional units can access them directly.

For the A and S registers, an intermediate level of registers exists. These registers are not accessible to the functional units, but act as a buffer for the primary registers. Block transfers of consecutive locations are possible between these registers and Central Memory so that the number of memory reference instructions required for scalar and address operands is greatly reduced. Data can then be moved from intermediate registers to the primary register when needed. The intermediate registers that support the A registers are referred to as intermediate address (B) registers. The intermediate registers that support S registers are referred to as intermediate scalar (T) registers.

Address Registers

Each CPU contains eight 32-bit A registers. The A registers serve a variety of applications but are primarily used as address registers for memory references and as index registers. They provide values for shift counts, loop control, and channel I/O operations and receive values of population count and leading zeros count. In address applications, A registers index the base address for scalar memory references and provide both a base address and an address increment for vector memory references.

Each CPU contains 64 B registers; each register is 32-bits wide. The B registers are used as intermediate storage for the A registers. Data is transferred between B registers and Central Memory, and between A and B registers. Typically, B registers contain data to be referenced repeatedly over a long time span, making it unnecessary to retain the data in either A registers or Central Memory. Examples of data stored in B registers are loop counts, variable array base addresses, and dimensions.

Address processing in the CRAY X-MP EA computer system operates in two modes: the X-mode and the Y-mode. In the X-mode, the A registers, B registers, and the address functional units are limited to 24 bits, as in earlier models of the CRAY X-MP computer systems. Only 1- and 2-parcel instructions run in the X-mode. In the Y-mode, the A registers, B registers, and address functional units run at a full 32-bit width and the instruction set is expanded to include 3-parcel instructions. Refer to "Instruction Differences Between the X-mode and Y-mode" later in this section for more information on these modes and instructions.

Scalar Registers

Each CPU contains 64 S registers; each register is 64-bits wide. The S registers are the principal scalar registers for a CPU. Scalar registers serve as the source and destination for scalar arithmetic and logical instructions. Scalar registers can also provide an operand for some vector operations.

Each CPU contains 64 T registers; each register is 64 bits wide. The T registers are used as intermediate storage for the S registers. Data is transferred between T registers and Central Memory and between T and S registers.

Vector Registers

Each CPU contains eight Vector (V) registers. Each V register consists of 64 elements; each element is 64-bits wide. The V registers serve as the source and destination for vector arithmetic and logical instructions. Successive elements from a V register enter a functional unit in successive CPs with a single instruction.

The effective length of a V register for any operation is controlled by the program-selectable Vector Length (VL) register. The VL register is a 7-bit register specifying the number of vector elements processed by vector instructions. The contents range from bits 0_8 through 77_8.

The Vector Mask (VM) register allows for the logical selection of particular elements of a vector. The VM register has 64 bits, each corresponding to a word element in a V register. The high-order bit of the VM register corresponds to element 0 of the V register, while the low-order bit corresponds to element 63. The mask is used with vector merge and test instructions to perform operations on individual elements.

Refer to "Vector Processing" later in this section for more specific information.

Functional Units

Instructions other than simple transmits or control operations are performed by specialized hardware known as functional units. Each unit implements an algorithm or a portion of the instruction set. Most functional units have independent logic and can operate simultaneously.

All functional units perform algorithms in a fixed time; delays are impossible after the operands are delivered to the unit (except those caused by memory conflicts). Functional units are fully segmented; this means a new set of operands for unrelated computation can enter a functional unit each CP even though the functional unit time can be more than 1 CP. Refer to "Pipelining and Segmentation" and "Functional Unit Independence" later in this section for more information on segmentation, pipelining, and functional unit independence.

There are four types of functional units: address, scalar, vector, and floating-point. The first three functional units function with one of the primary register types (A, S, and V) to support the address, scalar, and vector processing modes. The floating-point functional units, support either scalar or vector operations and accept operands from or deliver results to S or V registers. In addition, Central Memory can also act as a functional unit for vector operations.

Address Functional Units

Address functional units perform integer arithmetic on operands obtained from A registers and deliver the results to an A register (integer arithmetic is explained later in this section). The arithmetic is two's complement. The following list describes the two Address functional units.

- The Address Add functional unit performs integer addition and subtraction; subtraction is performed by using two's complement arithmetic. Overflow is not detected.
- The Address Multiply functional unit forms an integer product from two operands. No rounding is performed and overflow is not detected. The unit returns only the least significant bits of the product.

Scalar Functional Units

Scalar functional units perform operations on operands obtained from S registers and usually deliver the results to an S register (integer arithmetic is explained later in this section). The exception is the Population/Parity/Leading Zero Count functional unit, which delivers its result to an A register.

The Scalar Add, Scalar Shift, Scalar Logical, and Scalar Population/Parity/Leading Zero functional units are used exclusively with scalar operations and are described below. Three additional functional units are used for both scalar and vector operations. They are described in the "Floating-point Functional Units" subsection. The following list describes these four Scalar functional units:

- The Scalar Add functional unit performs integer addition and subtraction; subtraction is performed by using two's complement. Overflow is not detected.
- The Scalar Shift functional unit shifts the entire contents of an S register or shifts the contents of two linked S registers into a single resultant S register. Shifts are end-off with zero fill. Shift counts are obtained from an A register or from a field of the instruction.
- The Scalar Logical functional unit performs bit-by-bit manipulation of quantities obtained from S registers.

- The Scalar Population/Parity/Leading Zero functional unit counts the number of bits in an S register having a value of 1 in the operand, and then depending on the instruction issued, returns the value either as a population or population parity count to an A register. For the leading zero function, it counts the number of 0 bits preceding a 1 bit in the operand from left to right; the operand is obtained from an S register and the result is delivered to an A register.

Vector Functional Units

Vector functional units perform operations on operands obtained from one or two V registers or from a V register and an S register. The Vector Add and Logical functional units require two operands, while the Vector Shift and Population/Parity functional units require only one operand. Results from a Vector functional unit are delivered to a V register.

Successive operand pairs are transmitted each CP to a functional unit. The corresponding result emerges from the functional unit n CPs later, where n is the functional unit time and is constant for a given functional unit. The VL register determines the number of operands or operand pairs to be processed by a functional unit. Refer to "Special Features of the CRAY X-MP EA Computer System" later in this section for more information on vector processing, chaining, and other special vector-processing features.

The functional units described in this subsection are used exclusively with vector operations. Three functional units are used with both vector and scalar operations and are described in the following "Floating-point Functional Units" subsection. The following list describes the five Vector functional units.

- The Vector Add functional unit performs integer addition and subtraction for a vector operation and delivers the results to elements of a V register. Subtraction is performed by using two's complement arithmetic. Overflow is not detected.

- The Vector Shift functional unit shifts the entire contents of a V register element or the value formed from two consecutive elements of a V register. Shift counts are obtained from an A register. All shifts are end-off with zero fill.

- The Full Vector Logical functional unit performs a bit-by-bit manipulation of vector elements for specific instructions. The Full Vector Logical functional unit also performs vector register merge, compressed index, and logical operations associated with the vector mask instructions.

- The Second Vector Logical functional unit performs a bit-by-bit manipulation of the vector elements for specific instructions. The Second Vector Logical functional unit cannot perform vector register merge, compressed index, and logical operations associated with the vector mask instructions. A bit in the Exchange Package enables/disables the Second Vector Logical functional unit.

- The Vector Population/Parity functional unit counts the 1 bits in each element of the source V register; the result is the population count. This population count can be an odd or an even number, as shown by its low-order bit. The Vector Population Count instruction delivers the total population count to elements of the destination V register. The Vector Population Count Parity instruction delivers the low-order bit of the count to the destination V register.

Floating-point Functional Units

Three floating-point functional units perform floating-point arithmetic for scalar and vector operations. When a scalar instruction issues, operands are obtained from S register(s) and results are delivered to an S register. For most vector instructions, operands are obtained from pairs of V registers or from an S register and a V register. Results are delivered to a V register. An exception is the Reciprocal Approximation functional unit, which requires only one input operand. When a Floating-point functional unit is used for a vector operation, the general description of a vector functional unit applies. The following list describes the three floating-point functional units.

- The Floating-point Add functional unit performs addition or subtraction of operands in floating-point format. The final result is normalized even when operands are not normalized. (Refer to "Normalized Floating-point Numbers" later in this section for more information on normalized numbers.) Out-of-range exponents in operands (not in result values) are detected.

- The Floating-point Multiply functional unit issues instructions that provide for full- and half-precision multiplication of operands in floating-point format and for computing two minus a floating-point product for reciprocal iterations. The half-precision product is rounded; the full-precision product can be rounded or not rounded. This functional unit can also generate a 32-bit integer product when operating in X-mode or Y-mode.

  Input operands are assumed to be normalized. The Floating-point Multiply functional unit delivers a normalized result only if both input operands are normalized.

  Out-of-range exponents in operands are detected. If both operands have zero exponents, however, the result is an integer product, is not normalized, and is not out-of-range.

- The Reciprocal Approximation functional unit finds the approximate reciprocal of an operand in floating-point format. The input operand must be normalized. The high-order bit of the coefficient is not tested, but must be a 1. Out-of-range exponents are detected.

Functional Unit Operations

Functional units in a CPU perform logical operations, integer arithmetic, and floating-point arithmetic. Both types of arithmetic are performed in two's complement. The following subsections define the logical operations, the integer arithmetic, and the floating-point arithmetic used by the CRAY X-MP EA computer system.

Logical Operations

Scalar and vector logical units perform bit-by-bit manipulation of 64-bit quantities. Instructions are provided for forming logical products, sums, differences, equivalencies and merges.

A logical product is the AND function, which is shown below:

Operand 1:	1 0 1 0
Operand 2:	1 1 0 0
Result:	1 0 0 0

A logical sum is the inclusive OR function, which is shown below:

Operand 1:	1 0 1 0
Operand 2:	1 1 0 0
Result :	1 1 1 0

A logical difference is the exclusive OR function, which is shown below:

Operand 1:	1 0 1 0
Operand 2:	1 1 0 0
Result:	0 1 1 0

A logical equivalence is the exclusive NOR function,which is shown below:

Operand 1:	1 0 1 0
Operand 2:	1 1 0 0
Result:	1 0 0 1

The merge operation uses two operands and a mask to produce results as shown below. The bits of operand 1 pass where the mask bit is a 1. The bits of operand 2 pass where the mask bit is a 0.

Operand 1:	1 0 1 0 1 0 1 0
Operand 2:	1 1 0 0 1 1 0 0
Mask:	1 1 1 1 0 0 0 0
Result:	1 0 1 0 1 1 0 0

Integer Arithmetic

All integers, whether 24, 32, or 64 bits, are represented in the registers as shown in Figure 2-2. The Address Add and Multiply functional units perform 24-bit arithmetic in X-mode and 32-bit arithmetic in Y-mode (refer to "Instruction Differences Between the X-mode and Y-mode" later in this section for more information on these modes). The Scalar Add and Vector Add functional units perform 64-bit arithmetic.

Multiplication of two scalar (64-bit) integer operands is done using the Floating-point Multiply instruction and one of two multiplication methods. The method used depends on the magnitude of the operands and the number of bits available to contain the product. The following paragraphs explain the 24-bit integer multiply operation and the method used for operands greater than 24 bits.

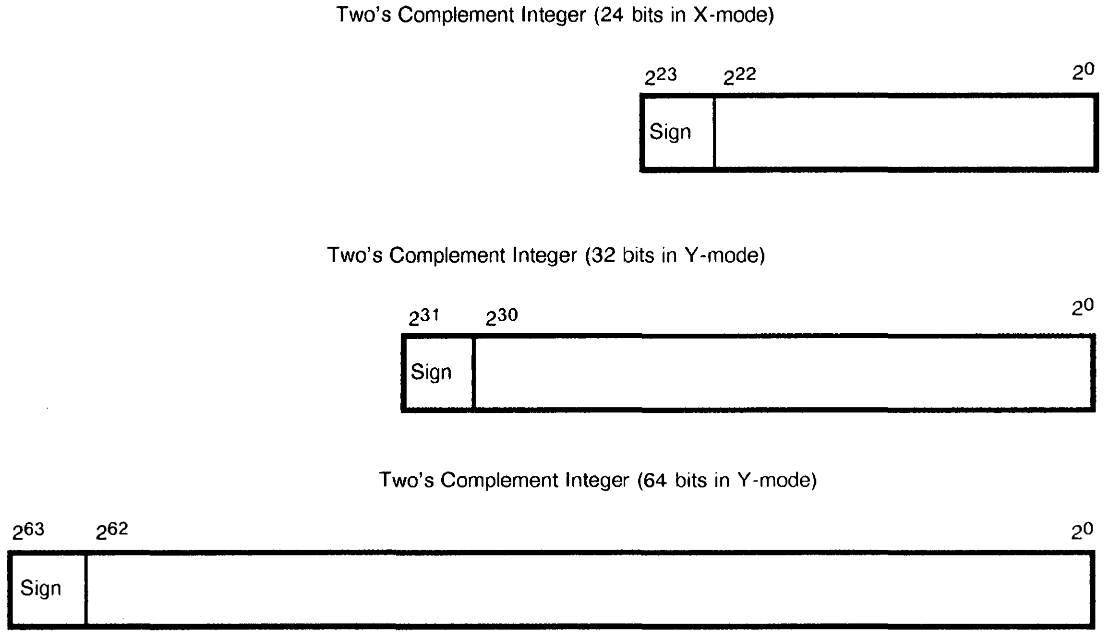

Two's Complement Integer (24 bits in X-mode)
2^{23}
2^{22}
2^0
Sign
Two's Complement Integer (32 bits in Y-mode)
2^{31}
2^{30}
2^0
Sign
Two's Complement Integer (64 bits in Y-mode)
2^{63}
2^{62}
2^0
Sign

Figure 2-2. Integer Data Formats

The Floating-point Multiply functional unit recognizes the condition under which both operands have zero exponents as a special case; it is treated as an integer multiplication operation, and a complete multiplication operation is performed with no truncation as long as the total number of bits in the two operands does not exceed 48-bit positions. To multiply two integer numbers together, set each operand's exponent equal to zero and place each 24-bit integer value in bit positions 2^{47} through 2^{24} of the operand's coefficient field. To ensure accuracy, the least significant 24 bits must be 0.

When the Floating-point Multiply functional unit has performed the operation, it returns the high-order 48 bits of the product as the result coefficient and leaves the exponent field as 0. The result is a 48-bit quantity in bit positions 2^{47} through 2^0; no normalization shift of the result is performed.

As shown in Figure 2-3, if operand 1 is 4_8 and operand 2 is 6_8, a 48-bit result of 30_8 is produced. Bit 2^{63} is subject to the usual rules for multiplying signs and the result is a sign-magnitude integer. The format of integers expected by both the hardware and software is two's complement, not sign-magnitude, therefore negative products must be converted to two's complement form.

The second multiplication method is used when the operands are greater than 24 bits in length. Multiplication is done by software forming multiple partial products and then shifting and adding the partial products.

A second integer multiplication operation performs a 32-bit multiply of the contents of S*j* register and the contents of V*k* to V*i*. The operands must be left-shifted before the operation begins. The operand contained in S*j* must be left-shifted 31_{10} places, leaving the operand in bit positions 2^{62} through 2^{31}; bit positions 2^{30} through 2^0 must be equal to 0 to ensure accuracy (refer to Figure 2-4).

The operand contained in V*k* must be left-shifted 16_{10} places, leaving the operand in bit positions 2^{47} through 2^{16}; bit positions 2^{15} through 2^0 must be equal to 0 to ensure accuracy. The result of the multiply is right justified into positions 2^{31} through 2^0, while positions 2^{32} through 2^{63} are zero filled.

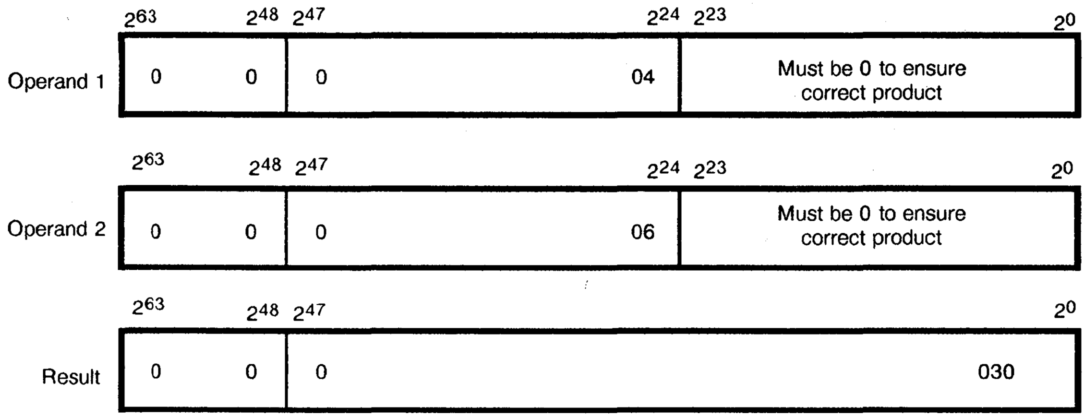


Figure 2-3. 24-bit Integer Multiply Performed in Floating-point Multiply Functional Unit

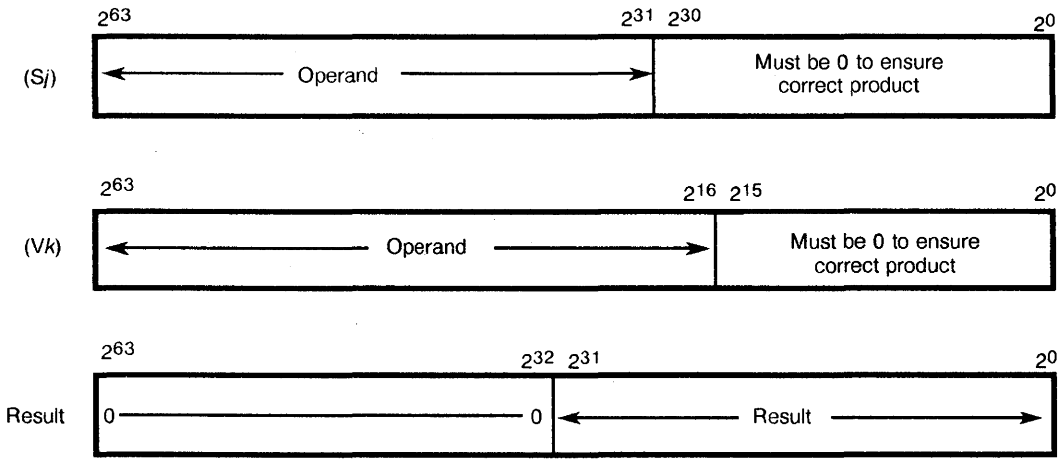


Figure 2-4. 32-bit Integer Multiply Performed in Floating-point Multiply Functional Unit

Although no integer division operation is provided, integer division can be carried out by converting the numbers to the floating-point format and then using the floating-point functional units. Refer to "Floating-point Division Algorithm" later in this section for more information.

Floating-point Arithmetic

Floating-point arithmetic is used by the scalar and vector instructions. The following subsections explain the floating-point data format, exponent ranges, normalized floating-point numbers, floating-point range errors, the floating-point addition, multiplication, and division algorithms, and double-precision numbers.

Floating-point Data Format

Floating-point numbers are represented in a standard format throughout the CPU; this format is shown in Figure 2-5. The format has three different fields: coefficient sign, exponent, and coefficient.

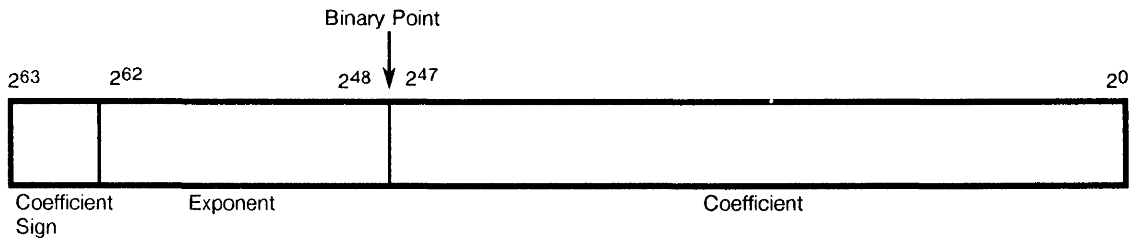


Figure 2-5. Floating-point Data Format

This format is a packed representation of a binary coefficient and an exponent (power of two). The coefficient sign is located in bit position 2^{63} and is separated from the rest of the coefficient. If this bit is equal to 0, the coefficient is positive; if this bit is equal to 1, the coefficient is negative. The exponent is represented as a biased integer number in bit positions 2^{62} through 2^{48}; each exponent is biased by 40000_8. Bit 2^{61} is the the sign of the exponent; a 0 indicates a positive exponent, while a 1 indicates a negative exponent. Bit 2^{62} is the bias of the exponent.

The coefficient is a 48-bit signed fraction; the sign of the coefficient is located in bit position 2^{63}. Because the coefficient is in sign-magnitude format, it is not complemented for negative values. A normalized floating-point number has a 1 in the 2^{47} bit position, while a not normalized floating-point number has a 0 in this bit position (normalized numbers are discussed in more detail later in this section).

Figure 2-6 and the following steps show the relationship between the bias, exponent, and coefficient.

To convert a floating-point number to its decimal equivalent, refer to the following example:

1. Subtract the bias from the exponent to get the integer value of the exponent:

$$\begin{array}{r} 40011_8 \\ \underline{-\,40000_8} \\ 11_8 = 9_{10} \end{array}$$

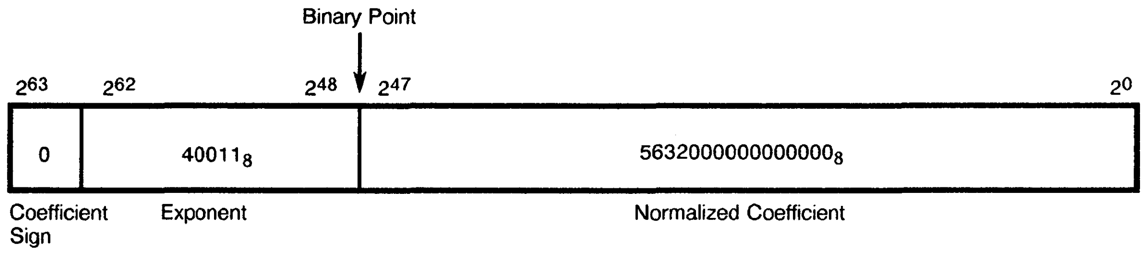


Figure 2-6. Internal Representation of Floating-point Number

2. Multiply the normalized coefficient by the power of 2 indicated in the exponent to get the result:

   $0.5634_8 \times 2^9 = 563.40_8 = 371.5_{10}$

A zero value or an underflow result is not biased and is represented as a word of all 0s. A negative 0 is not generated by any Floating-point functional unit, except if a negative 0 is one operand going into the Floating-point Multiply or Floating-point Add functional unit.

Exponent Ranges

The exponent portion of the floating-point format is represented as a biased integer in bits 2^{62} through 2^{48}. The bias that is added to the exponents is 40000_8, which represents an exponent of 2^0. Figure 2-7 shows the biased and unbiased exponent ranges.

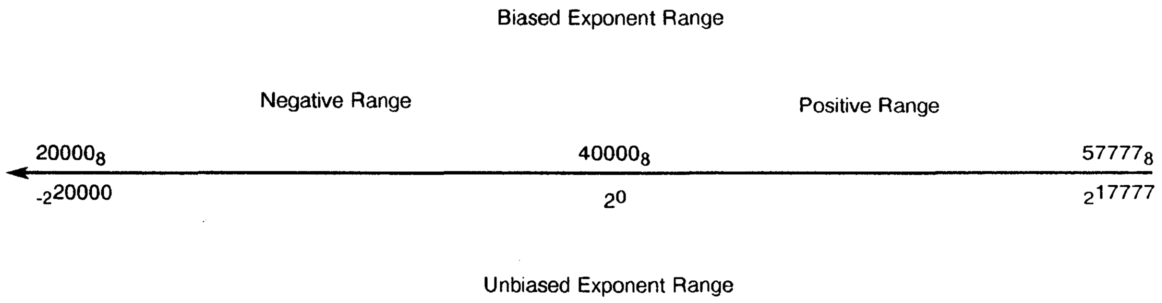


Figure 2-7. Biased and Unbiased Exponent Range

In decimal values, the floating-point format of the system allows the accurate expression of numbers to about 15 digits in the approximate range of 10^{-2466} through 10^{+2466}.

Normalized Floating-point Numbers

A nonzero floating-point number is normalized if the most significant bit of the coefficient, bit 2^{47}, is nonzero. This condition implies that the coefficient has been shifted as far left as possible and that the exponent has been adjusted accordingly, therefore a normalized floating-point number has no leading 0s in its coefficient. The exception is a normalized floating-point 0, which is all 0s.

When a floating-point number is created by inserting an exponent of 40060_8 and a 48-bit integer word into the coefficient, the result should be normalized before it is used in a floating-point operation. Normalization is accomplished by adding the not normalized floating-point operand to 0.

The Reciprocal Approximation functional unit must receive normalized numbers to produce correct results; using not normalized numbers will produce inaccurate results. The Floating-point Multiply functional unit does not require normalized numbers to get correct results; however, more accurate results occur when normalized numbers are used.

The Floating-point Add functional unit does not require normalized numbers to get correct results. The Floating-point Add functional unit does, however, automatically normalize all its results; not normalized floating-point numbers may be routed through this functional unit to take advantage of this process.

Floating-point Range Errors

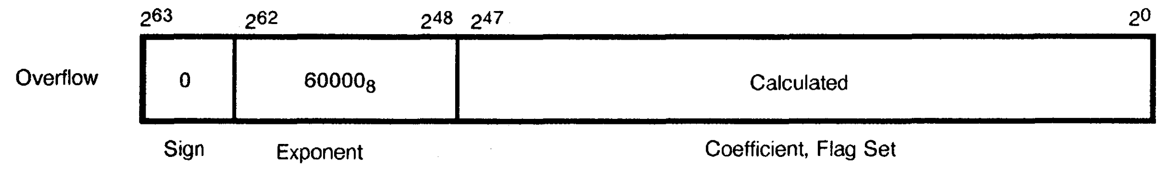
To make sure that the limits of the functional units are not be exceeded, a range check is made on the exponent of each floating-point operand for overflow and underflow conditions. In the Floating-point Add and Multiply functional units, bits 2^{61} and 2^{62} are checked; if both are equal to 1, the exponent is equal to or greater than 60000_8 and an overflow condition is detected. The calculated coefficient is reported, but the result exponent is set to 60000_8 and the Floating-point Error mode flag (IFP) is set (refer to Figure 2-8).

When an overflow condition is detected, an interrupt occurs only if the Interrupt on IFP flag is set in the Mode register and the system is not in Monitor mode. The IFP flag can be set or cleared by a user mode program.

To check for an underflow condition in the Floating-point Add and Multiply functional units, bits 2^{61} and 2^{62} are checked; if both are equal to 0, then the exponent is less than or equal to 17777_8 and an underflow condition is detected. No flag is set, but the exponent and coefficient are both set to 0 (refer to Figure 2-8).

In the Reciprocal Approximation functional unit, the exponent is complemented and the value of 2 is added before the operation proceeds. When the check is made in a reciprocal approximation operation, the exponent must be equal to or greater than 60002_8 to have an overflow condition occur. In this case, the calculated coefficient is reported, but bit 2^{47} is set to 0, the exponent is set to 60000_8, and the IFP is set (refer to Figure 2-9).

Again, because the reciprocal approximation operation complements and adds 2 to a floating-point number, the exponent must be less than or equal to 20001_8 for an underflow condition to occur. This underflow condition then causes an overflow condition on the original exponent. In this case, the calculated coefficient is reported, but bit 2^{47} is set to 0, the exponent is set to 60000_8, and the IFP is set (refer to Figure 2-9).



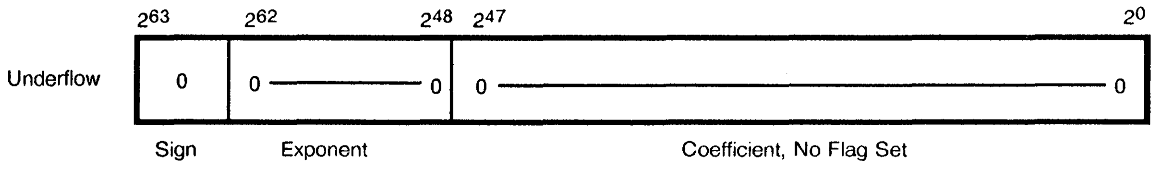
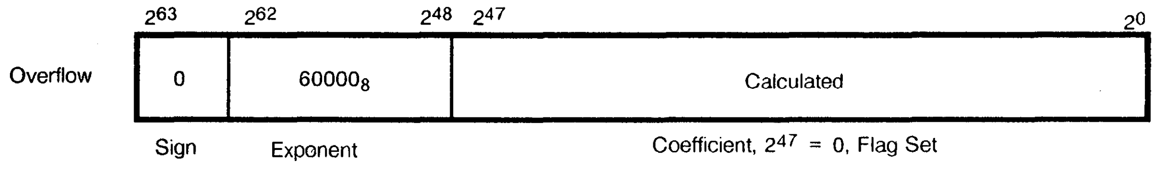


Figure 2-8. Floating-point Add and Multiply Range Errors



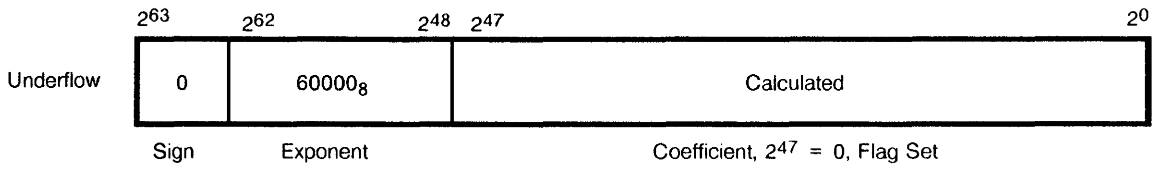


Figure 2-9. Floating-point Reciprocal Approximation Range Errors

Floating-point Addition Algorithm

Floating-point addition or subtraction is performed in a 49-bit register to allow for a sum that might carry into an additional bit position. The algorithm performs three operations: equalizing exponents, adding coefficients, and normalizing the results.

To equalize the exponents, the larger of the two exponents is retained. The coefficient of the smaller exponent is right-shifted by the difference of the two exponents, or until both exponents are equal. Bits shifted out of the register are lost; no roundup occurs. Because the coefficient is only 48 bits, any shift beyond 48 bits causes the smaller coefficient to become 0s.

After the two coefficients are equalized, they are added together. Two conditions are analyzed to determine whether an addition or subtraction operation occurs. The two conditions are the sign bits of the two coefficients and the type of instruction (an add or subtract) issued. The following list shows how the operation is determined.

- If the sign bits are equal and an addition instruction is issued, an addition operation is performed.
- If the sign bits are not equal and an addition instruction is issued, a subtraction operation is performed.
- If the sign bits are equal and a subtraction instruction is issued, a subtraction operation is performed.
- If the sign bits are not equal and a subtraction instruction is issued, an addition operation is performed.

The last operation performed is normalizing the results. To normalize the result, the coefficient is left-shifted by the number of leading 0s (the coefficient is normalized when bit 2^{47} is a 1). The exponent must also be decremented accordingly. If a carry across the binary point occurs during an addition operation, the coefficient is right-shifted by 1 and the exponent increases by 1. If a carry across the binary point occurs during a subtraction operation, an end-around carry occurs.

The normalization feature of the Floating-point Add functional unit can normalize any floating-point number; pair it with a zero operand and send them through the Floating-point Add functional unit.

A range check is performed on operands; refer to "Floating-point Range Errors" earlier in this section for more information on how the result is checked.

Floating-point Multiplication Algorithm

The Floating-point Multiply functional unit receives two floating-point operands from either an S or V register. The signs of the two operands are exclusive ORed, the exponents are added together, and the two 48-bit coefficients are multiplied together. If the coefficients are both normalized, multiplying them together produces a full product of either 95 or 96 bits. A 96-bit product is normalized as generated, while a 95-bit product requires a left-shift of one to generate the final coefficient. If the shift is done, the final exponent is reduced by 1 to reflect the shift.

Because the result register (an S or V register) can hold only 48 bits in the coefficient, only the upper 48 bits of the 96-bit result are used. The lower 48 bits are never generated. The following paragraphs describe the truncation process used to compensate for the loss of bits in the product. It assumes no shift was required to generate the final product; power of two designators are used.

The functional unit truncates part of the low-order bits of the 96-bit product. To adjust for this truncation, a constant is unconditionally added above the truncation. The average value of this truncation is 9.25×2^{-56}, which was determined by adding all carries produced by all possible combinations that could be truncated and dividing the sum by the number of possible combinations.

The effect of the truncation without compensation is at most a result coefficient 1 smaller than expected. With compensation, the results range from 1 too large to 1 too small in the 2^{-48} bit position, with approximately 99% of the values having zero deviation from what would have been generated had a full 96-bit product been present. The multiplication is commutative; that is, $A \times B = B \times A$.

Rounding is optional, while truncation compensation is not. The rounding method used adds a constant so that it is 50% high (0.25×2^{-48}; high) 38% of the time and 25% low (0.125×2^{-48}; low) 62% of the time, resulting in a near-zero average rounding error. In a full-precision rounded multiplication operation, 2 round bits are entered into the summation at bit positions 2^{-50} and 2^{-51} and allowed to multiply.

For a half-precision multiplication operation, round bits are entered into the summation at bit positions 2^{-32} and 2^{-31}. A carry resulting from this entry is allowed to multiply up and the 29 most significant bits of the normalized result are transmitted back.

The variations results from this truncation and rounding fall within the following range:

$$-0.23 \times 2^{-48} \text{ to } +0.57 \times 2^{-48}$$

or

$$-8.17 \times 10^{-16} \text{ to } +20.25 \times 10^{-16}$$

With a full 96-bit product and rounding equal to one-half the least significant bit, the following variation would be expected;

$$-0.5 \times 2^{-48} \text{ to } +0.5 \times 2^{-48}$$

Floating-point Division Algorithm

The CRAY X-MP EA computer system does not have a single functional unit that is dedicated to the division operation; the Floating-point Multiply and Reciprocal Approximation functional units together carry out the algorithm. The following paragraphs explain how the algorithm is determined and how it is used in the functional units.

Obtaining a quotient for two floating-point numbers involves two general steps. First to solve the equation A/B, the B operand is sent through the Reciprocal Approximation functional unit to obtain its reciprocal, 1/B. Then, this result is sent, along with the A operand to the Floating-point Multiply functional unit to obtain the product of $A \times 1/B$.

The steps involved in a division operation are not that general, however. The Reciprocal Approximation functional unit uses an application of Newton's method for approximating the real root of an arbitrary equation, $F(x) = 0$, to find reciprocals.

To find the reciprocal, the equation $F(x) = 1/x - B$ must be solved. To do this, a number A must be found so that $F(A) = 1/A - B = 0$. That is, the number A is the root of the equation $1/x - B = 0$. This method requires an initial approximation (or guess, which is shown as x_0 in Figure 2-10) sufficiently close to the true root (which is shown as x_t in Figure 2-10). x_0 is then used to obtain a better approximation; this is done by drawing a tangent line (line 1 in Figure 2-10) to the graph of $y = F(x)$ at the point $[x_0, F(x_0)]$.

The intercept of this tangent line becomes the second approximation, x_1. This process is repeated, using tangent line 2 to obtain x_2, and so on. The following iteration equation is derived from this process:

$$x_{(i+1)} = 2x_i - x^2_iB$$

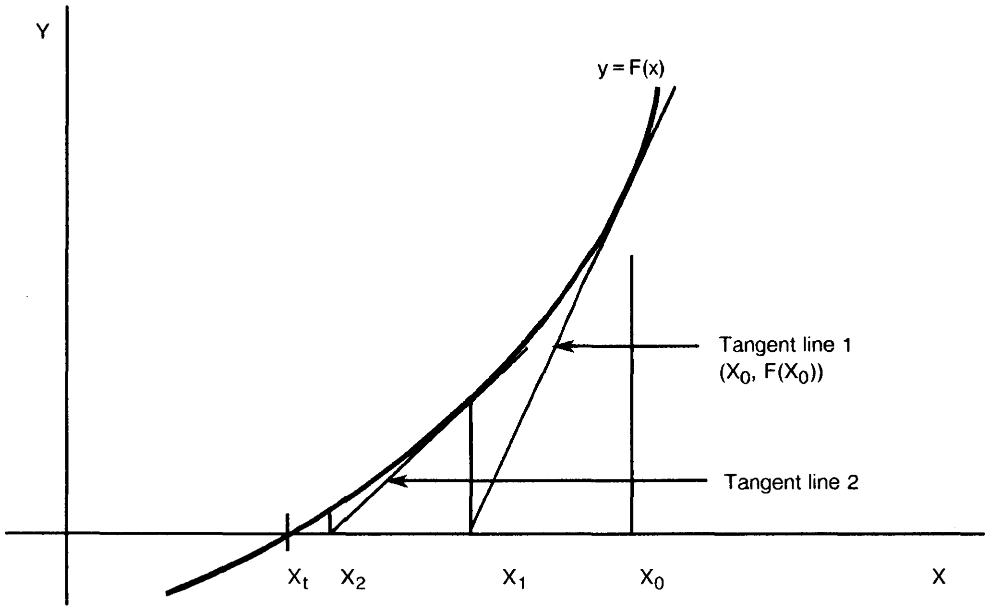


Figure 2-10. Newton's Method

In the equation, $x_{(x+1)}$ is the next iteration, x_i is the current iteration, and B is the divisor. Each $x_{(i+1)}$ is a better approximation than x_i to the true value, x_t. The exact answer is generally not obtained at once because the correction term is not exact. The operation is repeated until the answer becomes sufficiently close for practical use.

The CRAY X-MP EA mainframe uses this approximation technique based on Newton's method. A hardware look-up table provides an initial guess, x_0, to start the process. The following iterations are then calculated:

Iteration	Operation	Description
1	$x_1 = x_0(2 - x_0B)$	The first approximation is done in the Reciprocal Approximation functional unit and is accurate to 8 bits.
2	$x_2 = x_1(2 - x_1B)$	The second approximation is done in the Reciprocal Approximation functional unit and is accurate to 16 bits.
3	$x_3 = x_2(2 - x_2B)$	The third approximation is done in the Reciprocal Approximation functional unit and is accurate to 30 bits.

Iteration	Operation	Description
4	$x_4 = x_3(2 - x_3B)$	The fourth approximation is done in the Floating-point Multiply functional unit to calculate the correction term.

The Reciprocal Approximation functional unit calculates the first three iterations, while the the Floating-point Multiply functional unit calculates the fourth iteration. The fourth iteration uses a special instruction within the Floating-point Multiply functional unit to calculate the correction term. This iteration is used to increase accuracy of the Reciprocal Approximation functional unit's answer to full precision (the Floating-point Multiply functional unit can provide both full- and half-precision results). A fifth iteration should not be done.

The following example shows how the Floating-point Multiply functional unit is used to provide a full precision result, solving the equation S1/S2.

Step	Operation	Performed By
1	S3 = 1/S2	Reciprocal Approximation functional unit
2	S4 = [2 - (S3 * S2)]	Floating-point Multiply functional unit
3	S5 = S4 * S3	Floating-point Multiply functional unit using full precision; S5 now equals 1/S2 to 48-bit accuracy
4	S6 = S5 * S1	Floating-point Multiply functional unit using full-precision rounded

The reciprocal approximation in Step 1 is correct to 30 bits. By Step 3, it is accurate to 48 bits. This iteration answer is applied as an operand in a full-precision rounded multiplication operation (Step 4) to obtain a quotient accurate to 48 bits. Additional iterations may produce erroneous results.

Where 29 bits of accuracy are sufficient, the Reciprocal Approximation instruction is used with the half-precision multiplication operation to produce a half-precision quotient in only two operations, as shown in the following example.

Step	Operation	Performed By
1	S3 = 1/S2	Reciprocal Approximation functional unit
2	S6 = S1 * S3	Floating-point Multiply functional unit in half-precision

The 19 low-order bits of the half-precision results are returned as 0s with a rounding applied to the low-order bit of the 29-bit result.

The reciprocal iteration is designed for use once with each half-precision reciprocal generated. If the iteration performed by the Floating-point Multiply functional unit results in an exact reciprocal or if an exact reciprocal is generated by some other method, performing another iteration results in an incorrect final reciprocal.

Floating-point instructions include addition, subtraction, multiplication, and reciprocal approximation. The reciprocal approximation instructions can be used with a multiplication instruction sequence to perform a floating-point division operation.

The instruction set includes logical operations for AND, inclusive OR, exclusive OR, exclusive NOR, and a mask-controlled merge operation. Shift operations allow the manipulation of either 64-bit or 128-bit operands to produce 64-bit results. With the exception of address integer arithmetic, most operations are implemented in vector and scalar instructions.

The integer product is a scalar instruction designed for index calculation. Full indexing capability is possible throughout memory in either scalar or vector mode. The index can be positive or negative in either mode. Indexing allows matrix operations in vector mode to be performed on rows or on the diagonal as well as allowing conventional column-oriented operations.

Population and parity count instructions are provided for both vector and scalar operations. An additional scalar operation is the leading zero count.

Registers

Each CPU has three primary and two intermediate sets of registers. The primary sets of registers are the Address (A), Scalar (S), and Vector (V) registers. These registers are considered primary because Central Memory and the functional units can access them directly.

For the A and S registers, an intermediate level of registers exists. These registers are not accessible to the functional units, but act as a buffer for the primary registers. Block transfers of consecutive locations are possible between these registers and Central Memory so that the number of memory reference instructions required for scalar and address operands is greatly reduced. Data can then be moved from intermediate registers to the primary register when needed. The intermediate registers that support the A registers are referred to as intermediate address (B) registers. The intermediate registers that support S registers are referred to as intermediate scalar (T) registers.

Address Registers

Each CPU contains eight 32-bit A registers. The A registers serve a variety of applications but are primarily used as address registers for memory references and as index registers. They provide values for shift counts, loop control, and channel I/O operations and receive values of population count and leading zeros count. In address applications, A registers index the base address for scalar memory references and provide both a base address and an address increment for vector memory references.

Each CPU contains 64 B registers; each register is 32-bits wide. The B registers are used as intermediate storage for the A registers. Data is transferred between B registers and Central Memory, and between A and B registers. Typically, B registers contain data to be referenced repeatedly over a long time span, making it unnecessary to retain the data in either A registers or Central Memory. Examples of data stored in B registers are loop counts, variable array base addresses, and dimensions.

Address processing in the CRAY X-MP EA computer system operates in two modes: the X-mode and the Y-mode. In the X-mode, the A registers, B registers, and the address functional units are limited to 24 bits, as in earlier models of the CRAY X-MP computer systems. Only 1- and 2-parcel instructions run in the X-mode. In the Y-mode, the A registers, B registers, and address functional units run at a full 32-bit width and the instruction set is expanded to include 3-parcel instructions. Refer to "Instruction Differences Between the X-mode and Y-mode" later in this section for more information on these modes and instructions.

Scalar Registers

Each CPU contains 64 S registers; each register is 64-bits wide. The S registers are the principal scalar registers for a CPU. Scalar registers serve as the source and destination for scalar arithmetic and logical instructions. Scalar registers can also provide an operand for some vector operations.

Each CPU contains 64 T registers; each register is 64 bits wide. The T registers are used as intermediate storage for the S registers. Data is transferred between T registers and Central Memory and between T and S registers.

Vector Registers

Each CPU contains eight Vector (V) registers. Each V register consists of 64 elements; each element is 64-bits wide. The V registers serve as the source and destination for vector arithmetic and logical instructions. Successive elements from a V register enter a functional unit in successive CPs with a single instruction.

The effective length of a V register for any operation is controlled by the program-selectable Vector Length (VL) register. The VL register is a 7-bit register specifying the number of vector elements processed by vector instructions. The contents range from bits 0_8 through 77_8.

The Vector Mask (VM) register allows for the logical selection of particular elements of a vector. The VM register has 64 bits, each corresponding to a word element in a V register. The high-order bit of the VM register corresponds to element 0 of the V register, while the low-order bit corresponds to element 63. The mask is used with vector merge and test instructions to perform operations on individual elements.

Refer to "Vector Processing" later in this section for more specific information.

Functional Units

Instructions other than simple transmits or control operations are performed by specialized hardware known as functional units. Each unit implements an algorithm or a portion of the instruction set. Most functional units have independent logic and can operate simultaneously.

All functional units perform algorithms in a fixed time; delays are impossible after the operands are delivered to the unit (except those caused by memory conflicts). Functional units are fully segmented; this means a new set of operands for unrelated computation

The following is another method of computing division:

Step	Operation	Performed By
1	S3 = 1/S2	Reciprocal Approximation functional unit
2	S5 = S1 * S3	Floating-point Multiply functional unit
3	S4 = [2 - (S3 * S2)]	Floating-point Multiply functional unit
4	S6 = S4 * S5	Floating-point Multiply functional unit

With this method, the correction to reach a full-precision reciprocal is done after the number is multiplied by the half-precision reciprocal, rather than before the multiplication.

The coefficient of the reciprocal produced by this alternative method can be different by as much as 2×2^{-48} from the first method described for generating full-precision reciprocals. This difference can occur because one method can round up twice, while the other method may not round at all. One round up can occur while the correction is generated and the second round up can occur when the final quotient is being produced. Therefore, if the reciprocals are to be compared, use the same method each time the reciprocals are generated.

Double-precision Numbers

The CPU does not provide special hardware for performing double- or multiple-precision operations. Double-precision computations with 95-bit accuracy are available through software routines provided by CRI.

CPU CONTROL SECTION

The CPU's control section issues program instructions. Before program instructions can issue, exchange and instruction fetch sequences must occur. The following subsections describe the exchange mechanism (which includes defining both the Exchange Package and exchange sequence) and the instruction-fetch and instruction-issue sequences.

Exchange Mechanism

Each CPU uses an exchange mechanism for switching instruction execution from program to program. This exchange mechanism uses a CPU operation referred to as an exchange sequence and blocks of program parameters known as Exchange Packages.

Exchange Sequence

An exchange sequence must occur before a program can begin running. An exchange sequence performs two functions simultaneously. First, program parameters for the next program are loaded from Central Memory into registers in the CPU.

Second, parameters from the previous program are read from the registers and stored back into Central Memory.

The program parameters are held in an Exchange Package, which is described in the following subsections. The contents of the A and S registers are automatically saved in the Exchange Package; the contents of the B, T, V, VM, Shared Address (SB), Shared Scalar (ST), and Semaphore (SM) registers must be saved by software.

Exchange sequences may be initiated voluntarily by a deadstart sequence or program exit by the software, or automatically with an interrupt condition. All instructions previously issued are allowed to complete before the exchange sequence begins. An instruction fetch always follows an exchange sequence. Refer to "Instruction Fetch" later in this section for more information on this sequence.

Exchange Package

The Exchange Package is composed of a number of parameters, held in fields; these fields contain the contents of certain registers that are swapped during an exchange sequence. The following subsections define the fields of the Exchange Package.

Processor Number Field

The contents of the Processor Number (PN) field in the Exchange Package indicates which CPU performed the exchange sequence. This value is not initially stored in the Exchange Package before program execution; it is a constant inserted into the Exchange Package after the program has run.

Memory Error Data Fields

Memory error data consisting of six fields of information appears in the Exchange Package only if one of two conditions is met. The first condition is that the Interrupt on Correctable Memory error bit is set and a Correctable Memory error is detected; the second condition is that the Interrupt on noncorrectable Memory error bit is set and a noncorrectable Memory error is detected. Memory error data fields are described below.

- The Syndrome field identifies a SECDED error on a memory read or I/O channel.
- The Read Address Bank field identifies the bank where a memory read error occurred.
- The Read Error Type field identifies the type of memory or I/O error encountered; the error can be either correctable or noncorrectable.
- The Read Address Chip Select field identifies the chip on which a memory read error occurred.
- The Read Mode field determines the type of read mode in progress when a memory data error occured.

Program Address Register Field

The Program Address (P) register contents are stored in this field of the Exchange Package. The instruction at this location is the first instruction issued when this program begins to issue.

Address Base and Limit Fields

Four registers in the Exchange Package define a program's data range and instruction range in memory and has two benefits. First, all programs are relocatable. When a program is written, the programmer does not need to know where in memory the instruction and data fields will be located. Second, each program can have its memory access restricted. A program can be halted if it runs an instruction outside of its instruction range or if it reads or writes data outside of its data range. This is especially important where more than one program occupies memory at the same time; programs can be prevented from issuing instructions or operating on data that belongs to other programs. The four registers are described in the following list.

- The Instruction Base Address (IBA) register holds the base address of the user's instruction range. The IBA register determines where in memory an instruction fetch is made by adding the contents of the P register to the contents of the IBA register. The sum equals the absolute memory address for the fetch.

- The Instruction Limit Address (ILA) register holds the upper limit address of the user's instruction range. The ILA register determines the highest absolute address that can be accessed during an instruction fetch sequence. If this absolute address exceeds the ILA, a Program Range Error flag is set, which generates an interrupt.

- The Data Base Address (DBA) register holds the base address of the user's data range. The DBA register determines where in memory a program's data is located by adding the memory address generated by the instruction to the contents of the DBA register. The sum equals the absolute address for any memory read or write operation.

- The Data Limit Address (DLA) register holds the upper limit address of the user's data range. The DLA register determines the highest absolute memory address that a program can use for reading or writing data. If this absolute address exceeds the DLA, the memory reference is aborted. The Operand Range Error flag is set, which generates an interrupt if the Interrupt on Operand Range Error flag is set.

Exchange Address Register Field

The Exchange Address (XA) register specifies the first word address of a 16-word Exchange Package loaded by an exchange sequence. The register contains the high-order 8 bits of a 12-bit field specifying the address. The low-order bits of the field are always 0 because an Exchange Package must begin on a 16-word boundary. The 12-bit limit requires that the absolute address be in the lower 4,096 ($10{,}000_8$) words of memory. When an execution interval terminates, the exchange sequence exchanges the contents of the registers with the contents of the Exchange Package at the beginning address, or exchange address (XA), in memory.

Vector Length Register Field

The Vector Length (VL) register specifies the length of vector operations performed by vector instructions and the number of elements held by the V registers. The value in the VL register can be changed by software while a program is running.

Cluster Number Register Field

The Cluster Number (CLN) register determines which set of SB, SM, and ST registers the CPU can access. If the CLN register is 0, the CPU does not have access to any SB, SM, or ST register. The contents of the CLN register in all CPUs are also used to determine the condition necessary for a Deadlock interrupt command.

Flag Register Field

The Flag (F) register contains several flags which set the interrupt flag program issue by initiating an exchange sequence. The contents of the F register are stored along with the rest of the Exchange Package during the exchange sequence. The monitor program can then analyze the flags for the cause of the interrupt. Before the monitor program exchanges back, it must clear the flags in the F-register area of the Exchange Package. If any bit remains set, another exchange occurs immediately.

The F register contains the following flags:

- Interrupt from Internal CPU (ICP) flag: sets when another CPU issues instruction 0014*j*1 (not available on single-processor systems).
- Deadlock (DL) flag: sets when all CPUs in a cluster are holding issue on a Test and Set instruction.
- Programmable Clock Interrupt (PCI) flag: sets when the the Programmable Clock reaches a count of 0.
- MCU Interrupt (MCU) flag: sets when the Master I/O Processor (MIOP) sends this signal.
- Floating-point Error (FPE) flag: sets when a Floating-point Range error occurs in any of the floating-point functional units and the Interrupt on Floating-point Error flag is set.

- Operand Range Error (ORE) flag: set when a data reference is made outside the boundaries of the DBA and DLA registers and the Interrupt on Operand Range Error flag is set.
- Program Range Error (PRE) flag: set when an instruction fetch is made outside the boundaries of the IBA and ILA registers.
- Memory Error (ME) flag: set when a correctable or noncorrectable memory error occurs and the corresponding interrupt on Memory Error bit is set in the mode (M) register.
- I/O Interrupt (IOI) flag: set when a 6-Mbyte/s channel or a 1,000-Mbyte/s channel (100-Mbytes/s channel in single-processor models) completes a transfer.
- Error Exit (EEX) flag: if not in monitor mode, set by an Error Exit instruction.
- Normal Exit (NEX) flag: if not in monitor mode or if the Interrupt Monitor Mode bit is not set, set by a Normal Exit instruction.

Mode Register Field

The Mode (M) register contains user-selectable flags that dictate the execution of the program. The register also contains 2 status bits (Program State and Floating-point Error Status) that are set by software and hardware, respectively, during an exchange sequence.

The M register contains the following flags:

- Enable Second Vector Logical (ESVL) flag: when set, the Second Vector Logical functional unit can be used.
- Program State (PS) flag: this flag is set by the operating system to show whether a CPU concurrently processing a program with another CPU, is the master or slave in a multitasking situation.
- Floating-point Error Status (FPS) flag: when set, a Floating-point error occurred regardless of the state of the Interrupt on Floating-point Error flag.
- Bidirectional Memory (BM) flag: when set, block reads and writes can operate concurrently.
- Interrupt on Operand Range Error (IOR) flag: when set, this flag enables interrupts on Operand Address Range errors.
- Interrupt on Floating-point Error (IFP) flag: when set, this flag enables interrupts on floating-point errors.
- Interrupt on noncorrectable Memory Error (IUM) flag: when set, this flag enables interrupts on noncorrectable memory data errors.
- Interrupt on Correctable Memory Error (ICM) flag: when set, this flag enables interrupts on correctable memory data errors.

- Extended Addressing Mode (EAM) bit: when set, this bit indicates that 32-bit (Y-mode) addressing is taking place. When it is not set, it indicates that 24-bit (X-mode) addressing is taking place.
- Selected for External Interrupts (SEI) flag: when set, this CPU is preferred for I/O interrupts.
- Interrupt Monitor Mode (IMM) flag: when set, this flag enables all interrupts in monitor mode except DL, PCI, MCU, NEX, and IOI.
- Monitor Mode (MM) flag: when set, this flag inhibits all interrupts except memory errors, normal exit, and error exit.

Vector Not Used Field

The state of the Vector Not Used (VNU) bit in the Exchange Package indicates whether vector register instructions were issued during the execution intervals. If no vector register instructions were issued, the bit is set. If one or more of the vector register instructions were issued, the bit is not set.

Waiting for Semaphore Field

The state of the Waiting for Semaphore (WS) bit indicates that the CPU exchanged when a Test and Set instruction was holding in the Current Instruction Parcel (CIP) register.

A Registers Field

The current contents of all A registers are stored in this portion of the Exchange Package.

S Registers Field

The current contents of all S registers are stored in this portion of the Exchange Package.

Instruction Fetch

An instruction fetch operation loads program code from Central Memory to one of the instruction buffers. Each CPU has four instruction buffers; each holds 128 consecutive instruction parcels for a total of 512 parcels. Refer to "Instruction Formats" later in this section for more information on instruction formats and parcels. Instruction parcels are held in the buffers before being delivered to the instruction issue registers (refer to the following "Instruction Issue" subsection for a definition of these registers).

The contents of the Program Address (P) register determines when a fetch is made (refer to the following "Instruction Issue" subsection for a definition of the P register). If the P register is pointing to an instruction parcel not currently held in one of the instruction buffers, a fetch operation occurs.

A fetch operation always occurs following an exchange sequence. The instruction buffers are filled circularly as needed; when the P register counts 128 parcels, it reaches the end of the first instruction buffer. A second fetch occurs, filling the second instruction buffer, and so on, until all buffers are filled. If a program exceeds 512 parcels, the fifth fetch reloads the first instruction buffer.

Instruction Issue

Several registers are used for instruction issue. These registers receive instruction parcels from the instruction buffers, decode the instructions, check the availability of the necessary hardware, and issue the instruction. The following registers are used for instruction issue:

- Program Address (P) register: the P register selects an instruction parcel from one of the instruction buffers. This parcel is sent to the Next Instruction Parcel (NIP) register. Under normal circumstances, the P register increments sequentially as instructions are issued. However, branch instructions and exchange sequences can load the P register with any value.

- Next Instruction Parcel (NIP) register: the NIP register holds a parcel of program code before it enters the Current Instruction Parcel (CIP) register. Instruction decoding begins in this register.

- Current Instruction Parcel (CIP), Lower Instruction Parcel (LIP), and Lower Instruction Parcel 1 (LIP1) registers: the CIP register holds the instruction waiting to issue. If the instruction is a 2-parcel instruction, the CIP register holds the first parcel of the instruction and the LIP register holds the second parcel. If the instruction is a 3-parcel instruction, the CIP register holds the first parcel, the LIP register the second parcel, and the LIP1 the third parcel.

Programmable Clock

A Programmable Clock is a standard feature of the CRAY X-MP EA computer system. Each CPU has one Programmable Clock. This 32-bit clock can be loaded with a count value, which then decrements one count each CP. An interrupt signal is generated when the clock reaches a count of 0. These clocks allow the operating system to force interrupts signals at a particular time or frequency and to expedite the use of multitasking in programs. The clock frequency intervals vary for different models of the CRAY X-MP EA computer system.

Performance Monitor

The CRAY X-MP EA computer system contains a performance monitor. This monitor consists of several counters which track the following hardware-related events:

- The number of specific instructions that issue during program execution
- The number of hold issue conditions that occur during program execution
- The number of instruction fetches and memory conflicts that occur during program execution

The contents of the performance counters can be read into an S register. Using this information, programmers can increase the speed and enhance the efficiency of their programs.

Status Register

The Status register contains bits that indicate the operating modes of the CPU. These bits can be transferred to the high-order bit positions of a selected S register. The Status register bits indicate the following CPU states:

- Clustered, CLN not set to 0
- Noncorrectable Memory error occurred
- Correctable Memory error occurred
- Program State status
- Floating-point error occurred
- Floating-point interrupt enabled
- Operand range interrupt enabled
- Bidirectional memory enabled
- Processor number count (bits 0 through 1)
- Cluster number count (bits 0 through 2)

SPECIAL FEATURES OF THE CRAY X-MP EA COMPUTER SYSTEM

The CRAY X-MP EA computer system has several special features that enhance the parallel processing capabilities inherent in all Cray mainframes. Parallel processing can occur at the instruction level within one CPU and across multiple CPUs; the following subsections discuss parallel processing within a single CPU of a CRAY X-MP EA mainframe.

Parallel processing features within a single CPU include pipelining and segmentation, functional unit independence, and vector processing (vectorization). The first two features are inherent hardware features of the CRAY X-MP EA computer system. Vector processing is a feature that can be manipulated by a programmer to provide optimum throughput. These features are explained in later subsections.

Pipelining and Segmentation

Pipelining is defined as an operation or instruction beginning before a previous operation or instruction is complete. Pipelining is accomplished through the use of fully segmented hardware. Segmentation refers to the process whereby an operation is divided into a discrete number of sequential steps, or segments. Fully segmented hardware is designed to implement this segmentation by performing one segment of the operation during a single CP. At the beginning of the next CP, the partial results obtained are sent to the next segment of the hardware for processing the next step of the operation. During this CP, the previous hardware segment can process the next operation. If segmented hardware is not used, the whole operation or instruction has to finish before another starts.

In the CRAY X-MP EA computer system, segmented hardware includes all the hardware associated with exchange sequences, memory references, instruction fetch sequences, instruction issue sequences, and functional unit operations. The pipelining and segmentation features are critical to the execution of vector instructions.

Figure 2-11 shows how a set of elements is pipelined through a segmented Vector functional unit. In the first CP, element 1 of register V1 and element 1 of register V2 enters the first segment of the functional unit. During the next CP, the partial result is moved to the second segment of the functional unit, and element 2 of both Vector registers enters the first segment. This process continues each CP until all elements are completely processed. In this example, the functional unit is divided into five segments; the functional unit can process up to five different pairs of elements simultaneously. After 5 CPs, the first result leaves the functional unit and enters register V3; subsequent results are available at the rate of one per CP.

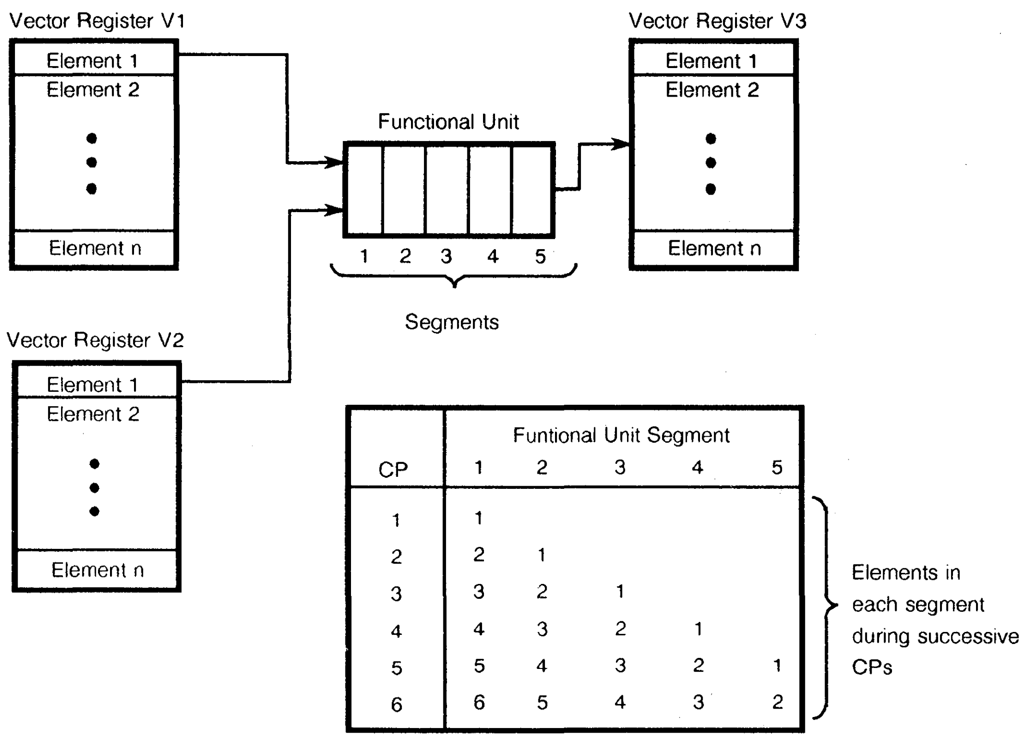
Functional Unit Independence

The specialized functional units in the CRAY X-MP EA computer system control the arithmetic, logical, and shift operations. Most units function independently, and any number of functional units can process instructions concurrently. This functional unit independence allows different operations, such as multiplication and addition to proceed in parallel.

For example, the equation, $A = (B + C) \times D \times E$, can run as follows. If operands B, C, D, and E are already loaded into the S registers, three instructions are generated for the equation: one that adds B and C, one that multiplies D and E, and one that multiplies the results of these two operations. The multiplication of D and E is issued first, followed by the addition of B and C. The addition and the multiplication proceed concurrently, and because the add takes less time to run than the multiply, the add and multiply complete at the same time. The add operation is essentially hidden because it occurs simultaneously with the multiply operation. The results of these two operations are then multiplied to obtain the final result.

Vector Processing

One of the most powerful features of the CRAY X-MP EA computer system is its vector processing capability. This feature increases processing speed efficiency by allowing an operation to be performed sequentially on a set (or vector) of operands, through the running of a single instruction. The following subsections describe vector processing, the advantages of using vector processing, and the types of vector instructions.



CP	Funtional Unit Segment 1	2	3	4	5
1	1				
2	2	1			
3	3	2	1		
4	4	3	2	1	
5	5	4	3	2	1
6	6	5	4	3	2

Figure 2-11. Segmentation and Pipelining Example

Definition of Vector Processing

Each CPU of the CRAY X-MP EA computer system contains V registers and a number of vector and floating-point functional units that perform vector operations. Refer to "Vector Registers", "Vector Functional Units", and "Floating-point Functional Units" in this section for more information on these registers and functional units.

A vector is an ordered set of elements; each is represented as a 64-bit word. A vector is distinguished from a scalar, which is a single 64-bit word. Examples of structures in Fortran that can be represented as vectors are one-dimensional arrays and rows, columns, and diagonals of multidimensional arrays. Vector processing occurs when arithmetic or logical operations are applied to vectors; it is distinguished from scalar processing because it operates on many elements rather than on one.

In vector processing, successive elements are provided in each CP, and as each operation is completed, the result is delivered to a successive element of the result register. The vector operation continues until the number of operations performed by the instructions equals the count specified by the Vector Length (VL) register.

Advantages of Vector Processing

In general, vector processing is faster and more efficient than scalar processing. Vector processing reduces overhead associated with maintenance of the loop control variable (for example, incrementing and checking the count). In many cases, loops processed as vectors reduce to a simple sequence of instructions without branching backwards. Vector instructions are usually the register-to-register type so that memory access conflicts are reduced. Finally, functional unit segmentation is exploited through vector processing, because results from the units can then be obtained at the rate of one result per CP.

Vectorization typically speeds up a code segment approximately by a factor of 10. If a code segment that previously accounted for 50% of a program's run time is vectorized, the overall run time is 55% of the original run time (50% for the unvectorized portion plus $0.1 \times 50\%$ for the vectorized portion). Vectorizing 90% of a program causes run time to drop to 19% of the original execution time.

Vector Chaining

The CRAY X-MP EA computer system allows a Vector register reserved for results to become the operand register of a succeeding instruction. This process, called chaining, allows a continuous stream of operands to flow through the vector registers and functional units. Even when a vector load operation pauses because of memory conflicts, chained operations may proceed as soon as data is available.

This chaining mechanism allows chaining to begin at any point in the result vector data stream. The number of concurrency separations in a chained operation depends on the relationship between the issue time of the chaining instruction and the result data stream. For full chaining to occur, the chaining instruction must have issued and be ready to use element 0 of the result at the same time element 0 arrives at the V register. Partial chaining occurs if the chaining instruction issues after the arrival of element 0.

Figure 2-12 shows how the results of four instructions are chained together. The sequence of instructions uses both the pipelining and segmentation features described in the previous subsection, along with the chaining mechanism to efficiently process the elements. The sequence of instructions performs the following operations:

1. Read a vector of integers from memory to the V0 register.
2. Add the contents of the V0 register to the contents of the V1 register and send the results to the V2 register.
3. Shift the results obtained in Step 2 and send the results to the V3 register.
4. Form the logical product of the shifted sum obtained in Step 3 with the V4 register, and send the results to the V5 register.

Elements are loaded into the V0 register. As soon as the first element arrives from memory into the V0 register, it is added to the first element of the V1 register. Subsequent elements are pipelined through the segmented functional unit so that a continuous stream of results is sent to the destination register, which is Vector register V2. As soon as the first element arrives at V2, it becomes the operand for the shift operation. The results are sent to V3, which immediately becomes the source of one of the operands necessary for the logical operation between V3 and V4. The results of the logical operation are then sent to Vector register V5.

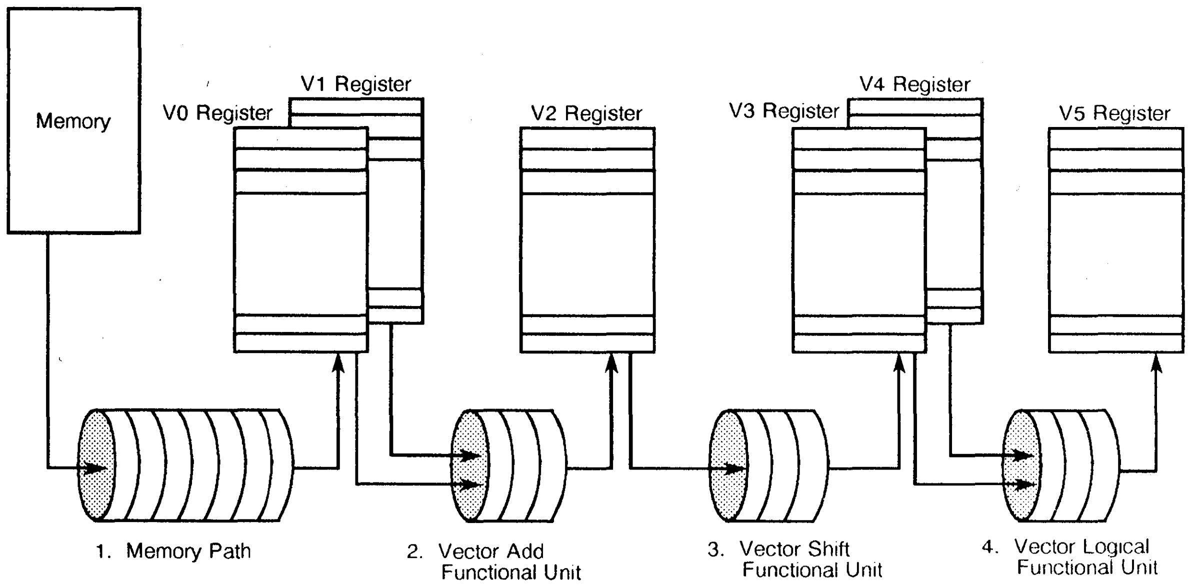

Memory
V0 Register
V1 Register
V2 Register
V3 Register
V4 Register
V5 Register
1. Memory Path
2. Vector Add Functional Unit
3. Vector Shift Functional Unit
4. Vector Logical Functional Unit

Figure 2-12. Vector Chaining Example

Types of Vector Instructions

The instructions that operate on vectors can be divided into four types:

- Vector to vector operand instructions that obtain operand(s) from one or two V registers and enter results into another V register
- Vector to scalar operand instructions that obtain one operand (a constant) from an S register and one operand from a V register and enter results in another V register
- Vector memory instructions that load (read) or store (write) elements to memory
- Vector instructions that set/read the Vector Mask (VM) register or set/read the Vector Length (VL) register

Refer to "Functional Instruction Summary" later in this section for more information on the specific instructions.

Figure 2-13 shows how the data flows when these instructions issue. Successive operands or operand pairs are transmitted from V*j* and/or V*k* to the segmented functional unit each CP. Corresponding results emerge from the functional unit n CPs later where n is a constant for a given functional unit and is called the functional unit time. Results are then entered into result register V*i*. Contents of the VL register determine the number of operand pairs processed by the functional unit.

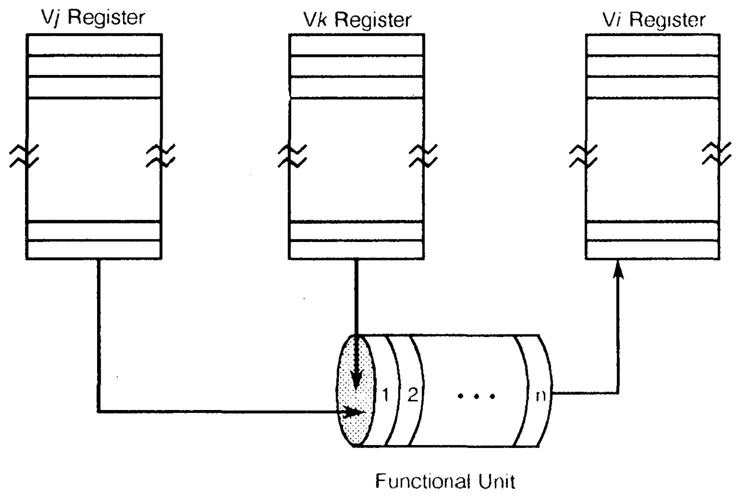


Figure 2-13. Vector-vector Operand Instructions

The vector-scalar operand instructions obtain one operand from an S register and one from a V register (refer to Figure 2-14). A copy of the S register is transmitted to the functional unit with each V-register operand. Refer to "Functional Instruction Summary" later in this section for more information on the specific instructions. Vector memory instructions transmit data between memory and the V registers (refer to Figure 2-15). A path between memory and the V registers is considered a functional unit for timing considerations. Refer to "Functional Instruction Summary" later in this section for more information on the instructions.

Memory access and vector processing are closely related. A special gather/scatter mechanism has been developed on the CRAY X-MP EA computer system to allow access to memory for vector operations in cases where vectorization would otherwise not be possible.

Most vector memory instructions access memory addresses with a fixed increment value. The Gather and Scatter instructions use two vector registers to gather or scatter elements randomly throughout memory. The first vector register contains the data and the second vector register is used as an index to gather or scatter the data from/to memory locations that cannot be expressed with a fixed increment.

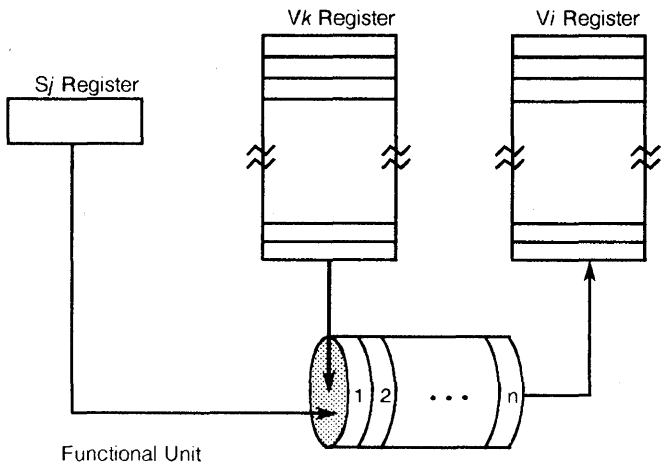


Figure 2-14. Vector-scalar Operand Instructions

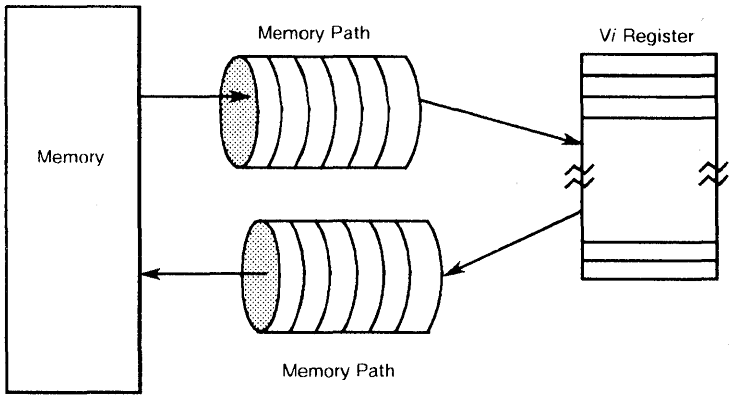


Figure 2-15. Vector Memory Instructions

Figure 2-16 shows an example of the Gather instruction. The Gather instruction transfers the contents of nonsequential memory locations to elements of a V register. In the example the VL register is set to 4, resulting in a transfer of 4 elements. The Gather instruction adds the contents of A0 to the contents of each element of the index V register (V0) to form a memory address. The contents of that address are then stored in the result V register (V1). Since A0 = 100 and V0 element 0 = 4, the contents of address 104 is stored in V1 element 0. Similarly, A0 + V0 element 1 = 102, and the contents of memory location 102 is stored in V1 element 1. This process continues until the number of elements transferred equals the VL count.

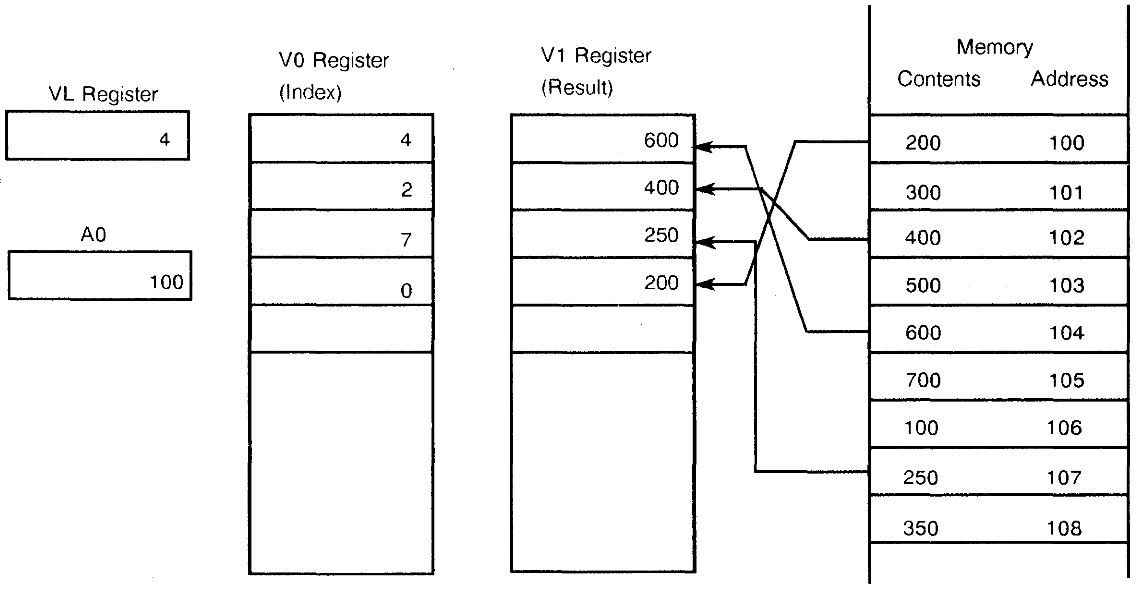


Figure 2-16. Gather Instruction Example

Figure 2-17 shows an example of the Scatter instruction. The Scatter instruction transfers elements of a V register to nonsequential memory locations. In the example, the VL register is set to 4, resulting in a transfer of 4 elements. The Scatter instruction adds the contents of the A0 register to the contents of each element of the index V register (V0) to form a memory address. An element of the V1 register is stored at the resulting memory address. Since A0=100 and V0 element 0=4, the contents of V1 element 0 is stored in address 104. Similarly, A0 + V0 element 1 = 102, and the contents of V1 element 1 is stored in memory location 102. This process continues until the number of elements transferred equals the VL count.

The fourth group of instructions set the VM register or read/set the VL register. (Refer to "Functional Instruction Summary" later in this section for more information on the specific instructions.) The VM register has 64 bits, each corresponding to a word element in a V register. The high-order bit of the VM register corresponds to element 0 of the V register, while the low-order bit corresponds to element 63. The mask is used with vector merge and test instructions to perform operations on individual elements.

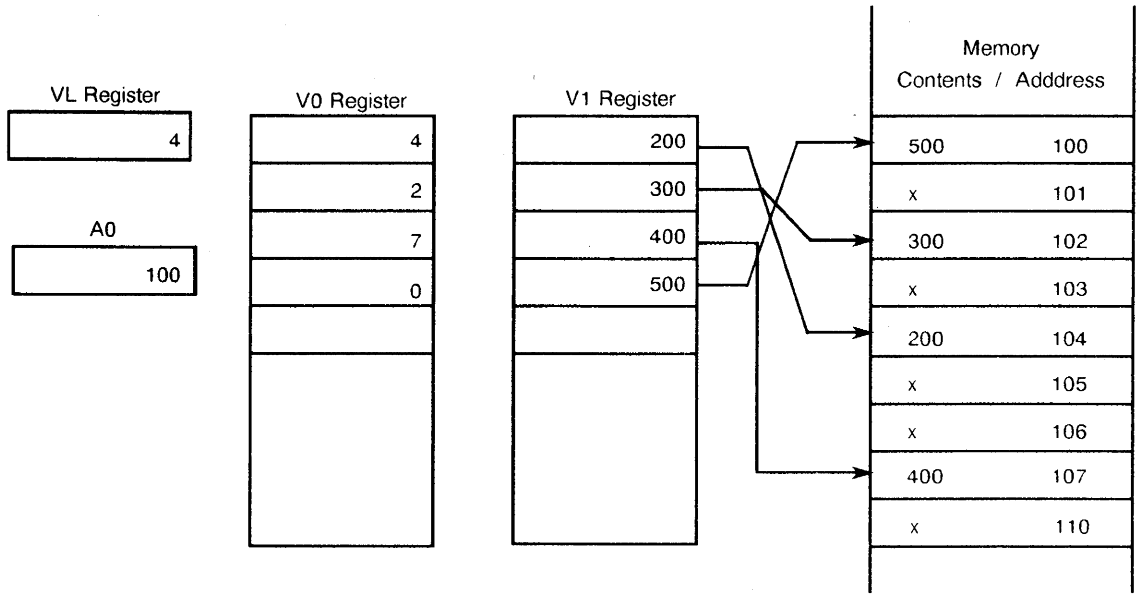


Figure 2-17. Scatter Instruction Example

The VM instructions include four compressed index instructions. These instructions test for zero, nonzero, positive, and negative elements, and generate a vector mask at the same time. Figure 2-18 shows an example of a compressed index instruction.

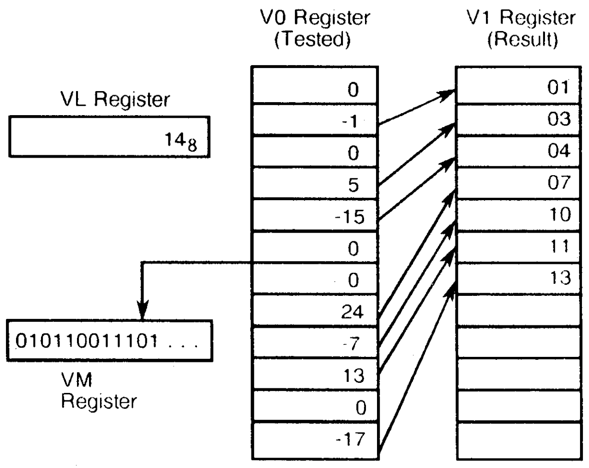


Figure 2-18. Compressed Index Example

In the example, the elements in V0 are individually tested for a non-zero status; if the element is 0, a 0 is entered in the VM register. If the element is non-zero, a 1 is entered in the VM register and the index of the non-zero elements is loaded into register Vl. This process continues until the number of elements specified in the VL register has been tested.

CPU INSTRUCTIONS

CPU INSTRUCTIONS

The following subsections explain the instruction formats, instruction differences between the X-mode and the Y-mode, and special register values used by the CRAY X-MP EA computer system. The X-mode uses the instruction set from earlier CRAY X-MP models, while the Y-mode uses the upward-compatible CRAY Y-MP instruction set. The ability to move between X-mode and Y-mode is software selectable. A CPU instruction summary is also included.

INSTRUCTION FORMATS

Instructions can be 1-parcel (16-bit), 2-parcels (32-bit), or 3-parcels (48-bit, Y-mode only) long. Instructions are packed 4 parcels per word and parcels are numbered 0 through 3 from left to right. Any parcel position can be addressed in branch instructions. A 2- or 3-parcel instruction begins in any parcel of a word and can span a word boundary. For example, a 2-parcel instruction beginning in parcel 3 of a word ends in parcel 0 of the next word. No padding to word boundaries is required. Figure 2-19 shows the general format of instructions.

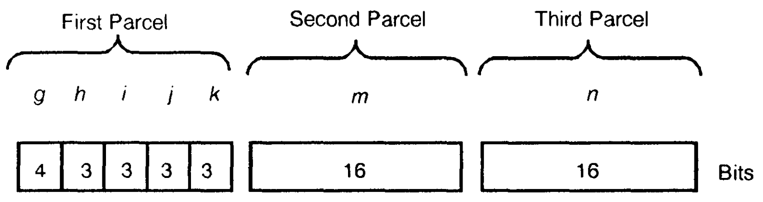


Figure 2-19. General Format for Instructions

Five variations of this general format use the fields differently. The formats of the following variations are described in the following subsections.

- 1-parcel instruction format with discrete *j* and *k* fields
- 1-parcel instruction format with combined *j* and *k* fields
- 2-parcel instruction format with combined *j*, *k*, and *m* fields
- 2-parcel instruction format with combined *i*, *j*, *k*, and *m* fields
- 3-parcel instruction format with combined *m* and *n* fields

1-parcel Instruction Format with Discrete *j* and *k* Fields

The most common of the 1-parcel instruction formats uses the *i*, *j*, and *k* fields as individual designators for operand and result registers (refer to Figure 2-20). The *g* and *h* fields define the operation code, the *i* field designates a result register, and the *j* and *k* fields designate operand registers. Some instructions ignore one or more of the *i*, *j*, and *k* fields.

The following types of instructions use this format:

- Arithmetic
- Logical
- Double Shift
- Floating-point Constant

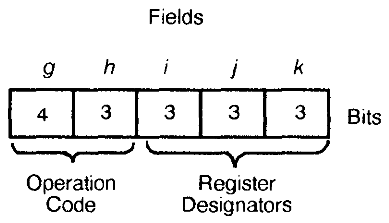


Figure 2-20. 1-parcel Instruction Format with Discrete *j* and *k* Fields

1-parcel Instruction Format with Combined *j* and *k* Fields

Some 1-parcel instructions use the *j* and *k* fields as a combined 6-bit field (refer to Figure 2-21). The *g* and *h* fields contain the operation code, and the *i* field is generally a destination register. The combined *j* and *k* fields generally contain a constant or a B or T register designator. The Branch instruction 005 and the following types of instructions use the 1-parcel instruction format with combined *j* and *k* fields:

- Constant
- B and T register block memory transfer
- B and T register data transfer
- Single shift
- Mask

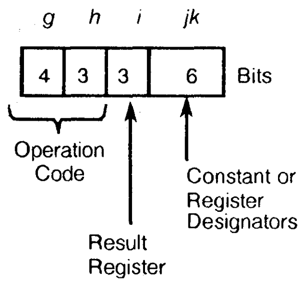
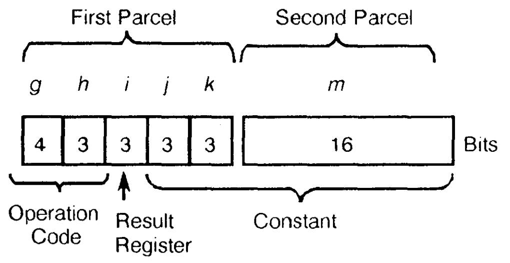


Figure 2-21. 1-parcel Instruction Format with Combined *j* and *k* Fields

2-parcel Instruction Format with Combined *j*, *k*, and *m* Fields

The format for a 22-bit immediate constant uses the combined *j*, *k*, and *m* fields to hold the constant. The 7-bit combination of the *g* and *h* fields contains an operation code and the 3-bit *i* field designates a result register. The instruction using this format transfers the 22-bit *jkm* constant to an A or S register.

The instruction format used for scalar memory transfers also requires a 22-bit *jkm* field for address displacement. This format uses the 4-bit *g* field for an operation code, the 3-bit *h* field to designate an address index register, and the 3-bit *i* field to designate a source or result register. Figure 2-22 shows the two general applications for the 2-parcel instruction format with combined *j*, *k*, and *m* fields.



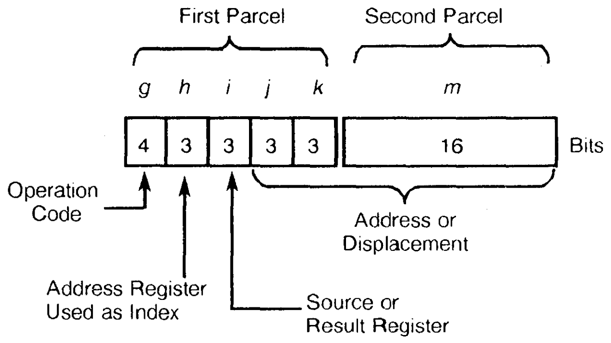


Figure 2-22. 2-parcel Instruction Format with Combined *j*, *k*, and *m* Fields

2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields

This 2-parcel format uses the combined *i*, *j*, *k*, and *m* fields to contain a 24-bit address that allows branching to an instruction parcel (refer to Figure 2-23). A 7-bit operation code (*gh*) is followed by an *ijkm* field. The high-order bit of the *i* field is equal to 0.

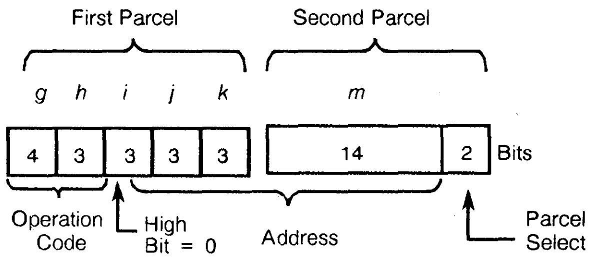


Figure 2-23. 2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields

The 2-parcel format for a 24-bit immediate constant (refer to Figure 2-24) uses the combined *i*, *j*, *k*, and *m* fields to hold the constant. This format uses the 4-bit *g* field for an operation code and the 3-bit *h* field to designate the result address register. The high-order bit of this *i* field is set.

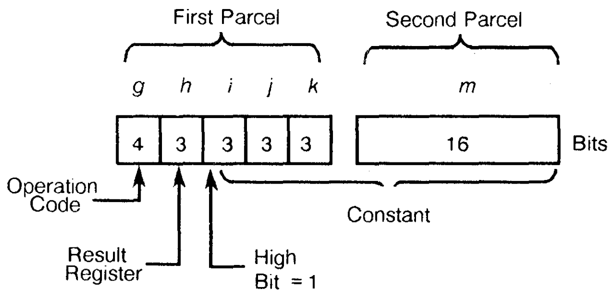


Figure 2-24. 2-parcel Instruction Format for a 24-bit Immediate Constant with Combined *i*, *j*, *k*, and *m* Fields

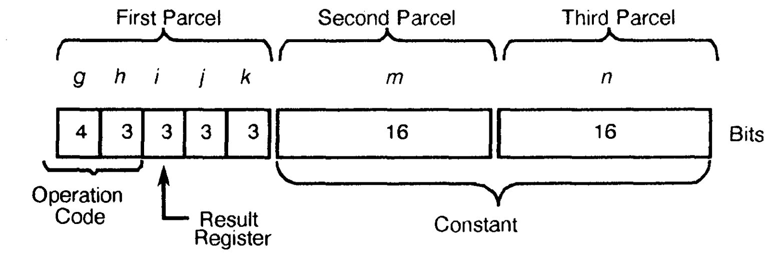
3-parcel Instruction Format with Combined *m* and *n* Fields

The format for a 32-bit immediate constant uses the combined *m* and *n* fields to hold the constant. The 7-bit *g* and *h* fields contain an operation code, and the 3-bit *i* field designates a result register; the *j* and *k* fields are a constant 0. The instruction using this format transfers the 32-bit *mn* constant to an A or S register.

Note: The *m* field of the 3-parcel instruction contains bits 2^0 through 2^{15} of the expression, while the *n* field contains bits 2^{16} through 2^{31} of the expression. When the instruction is assembled, the *mn* field is "reversed" and actually appears as the *nm* field when used as an expression.

The format used for scalar memory transfers also requires a 32-bit *mn* field for address or displacement. This format uses the 4-bit *g* field for an operation code, the 3-bit *h* field to designate an address index register, and the 3-bit *i* field to designate a source or result register.

Figure 2-25 shows the two general applications for the 3-parcel instruction format with combined *m* and *n* fields.



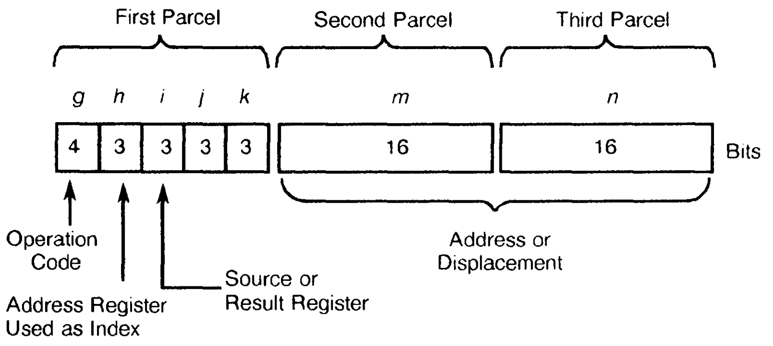


Figure 2-25. 3-parcel Instruction Format with Combined *m* and *n* Fields

INSTRUCTION DIFFERENCES BETWEEN THE X-MODE AND Y-MODE

The CRAY X-MP EA computer system runs either of two instruction modes: the X-mode and the Y-mode. In the Y-mode, the instruction set is expanded to include 3-parcel instructions (refer to Table 2-1), and the A registers, B registers, and the address functional units operate at a full 32-bit width. These 3-parcel instructions run only if the system is operating in Y-mode; use of these instructions while in X-mode produces errors. The program range remains 4-million words in both the X-mode and Y-mode.

Table 2-1. CRAY X-MP EA Computer System 3-parcel Instructions

CAL Syntax	Octal Code
A*i exp*	020*i*00*mn*
A*i exp*	021*i*00*mn*
S*i exp*	040*i*00*mn*
S*i exp*	041*i*00*mn*
A*i exp*, A*h*	10*hi*00*mn*
exp, A*h* A*i*	11*hi*00*mn*
S*i exp*, A*h*	12*hi*00*mn*
exp, A*h* S*i*	13*hi*00*mn*

The X-mode can be selected by resetting the Extended Addressing Mode bit in the Exchange Package. In this mode, the system runs only the X-mode (1- and 2-parcel) instruction set. The upper 8 bits of the 32-bit registers and 32-bit results are discarded, leaving the operation exactly the same as the earlier model 24-bit CRAY X-MP computer system results.

All instructions operate the same as in the earlier models of CRAY X-MP computer systems, except those listed in Table 2-2. For a complete explanation of these and all other instructions, refer to the manuals listed in Section 6 under "Software Publications".

Table 2-2. X-mode/Y-mode Instruction Differences

Instruction	X-Mode	Y-mode	Comments
01*hijkm*	A*h exp*	N/A	Not allowed in Y-mode
0014*j*1	SIPI A*j*	SIPI A*j*	Change is necessary due to more CPUs available
0014*j*3	CLN A*j*	CLN A*j*	Change is necessary due to more clusters in CRAY Y-MP mainframe
166*ijk*	V*i* S*j**IV*k*	V*i* S*j**V*k*	Runs differently in the X-mode than in the Y-mode

SPECIAL REGISTER VALUES

If the S0 and A0 registers are referenced in the *h*, *j*, or *k* fields of certain instructions, the contents of the respective register are not used; instead, a special operand is generated. The special operand is available regardless of existing A0 or S0 reservations (and in this case is not checked). This use does not alter the actual value of the S0 or A0 register. If S0 or A0 is used in the *i* field as the operand, the actual value of the register is provided. CAL issues a caution-level error message for A0 or S0 when 0 does not apply to the *i* field. Table 3-3 shows the special register values.

Table 3-3. Special Register Values

Field	Operand Value
A*h*, *h* = 0	0
A*i*, *i* = 0	(A0)
A*j*, *j* = 0	0
A*k*, *k* = 0	1
S*i*, *i* = 0	(S0)
S*j*, *j* = 0	0
S*k*, *k* = 0	2^{63}

MONITOR MODE INSTRUCTIONS

The monitor mode instructions (channel control, set Real-time Clock, and Programmable Clock interrupts) perform specialized functions that are useful to the operating system. These instructions run only when the CPU is operating in monitor mode. If a monitor mode instruction issues while the CPU is not in monitor mode, it is treated as a no-operation.

SPECIAL CAL SYNTAX FORMS

The CAL instruction set has special forms of symbolic instructions. Because of this expansion, certain machine instructions can be generated from two or more different CAL instructions. Any of the operations performed by special instructions can be performed by instructions in the basic set.

For example, both of the following CAL instructions generate instruction 002000, which enters a 1 into the VL register:

```
VL A0
VL 1
```

The first instruction is the basic form of the Enter VL instruction, which takes advantage of the special case where (A*k*)=1 if *k*=0; the second instruction is a special syntax form providing the programmer with a more convenient notation for the special case.

In several cases, a single CAL syntax can generate several different machine instructions. These cases provide for entering the value of an expression into an A register or an S register, or for shifting S register contents. The assembler determines which instruction to generate from characteristics of the expression.

Instructions having a special syntax form are identified in the instruction summary later in this section.

CPU INSTRUCTION SUMMARY

This subsection introduces and summarizes all instructions used by the CRAY X-MP EA mainframe. The instructions are summarized two ways: by the functional unit that executes the instruction and by the function the instruction performs.

The following instruction summaries use the acronyms and abbreviations that were defined in previous sections. A glossary is provided at the end of this manual; all acronyms and abbreviations are defined there.

In some instructions, register designators are prefixed by the following letters that have special meaning to the assembler. These letters can be in either uppercase or lowercase (case insensitive). The letters and their meanings are listed as follows.

Letter	Description
F	Floating-point operation
H	Half-precision floating-point operation
I	Reciprocal iteration
P	Population count
Q	Parity count
R	Rounded floating-point operation
Z	Leading-zero count

The following list defines some of the notations used in the instruction set.

Character	Operation
+	Arithmetic sum of specified registers
-	Arithmetic difference of specified registers
*	Arithmetic product of specified registers
/	Reciprocal approximation
#	Use one's complement
>	Shift value or form mask from left to right
<	Shift value or form mask from right to left
&	Logical product of specified registers
!	Logical sum of specified registers
\	Logical difference of specified registers

An expression (*exp*) occupies the *jk, ijk, jkm, ijkm,* or *ijkmn* field. The *h, i, j,* and *k* designators indicate the field of the machine instruction into which the register designator constant or symbol value is placed.

Functional Units Instruction Summary

Instructions other than simple transmits or control operations are performed by specialized hardware known as functional units. The following list summarizes the instructions performed by each of the functional units.

Functional Unit	Instructions
Address Add (Integer)	030, 031
Address Multiply (Integer)	032
Scalar Add (Integer)	060, 061
Scalar Logical	042-051
Scalar Shift	052, 057
Scalar Pop/Parity/	026
Leading Zero	027
Vector Add (Integer)	154-157
Vector Logical	140-147, 175
Second Vector Logical	140-145
Vector Shift	150, 152
Vector Pop/Parity	174*ij*1, 174*ij*2
Floating-point Add	062, 063, 170-173
Floating-point Multiply	064-067, 160-167
Floating-point Reciprocal	070, 174*ij*0
Memory (Scalar)	100-130
Memory (Vector)	176, 177

Functional Instruction Summary

This subsection summarizes the instructions by the functions they perform. Included is a brief general description of the function of each group of instructions; the machine instruction, the CAL syntax, and a description is then listed. For more information on these instructions, refer to the manuals listed in Section 6 under "Software Publications".

Note: The following footnotes are used throughout the instruction summary:

Footnote	Description
1	Privileged to monitor mode
2	Special Syntax Mode
3	Not supported by CAL Version 2
4	Generated depending on the value of *exp*
5	X-mode only
6	Y-mode only

Register Entry Instructions

The register entry instructions transmit values such as constants, expression values, or masks directly into registers.

Transfers Into A Registers

The following instructions transmit values into the A registers.

Machine Instruction	CAL Syntax	Description
01*hijkm*[5]	A*h* *exp*	Transmit *exp* to A*h* ($i^2=1$)
020*ijkm*[4,5] or 021*ijkm*[4,5]	A*i* *exp*	Transmit *exp* into A*i* (020) or Transmit one's complement of *exp* into A*i* (021)
020*i*00*mn*[4,6] or 021*i*00*mn*[4,6]	A*i* *exp*	Transmit *exp* into A*i* (020) or Transmit one's complement of *exp* into A*i* (021)
022*ijk*[4]	A*i* *exp*	Transmit *exp*=*jk* to A*i*
031*i*00[2]	A*i* -1	Transmit -1 into A*i*

Transfers Into S Registers

The following instructions transmit values into the S registers.

Machine Instruction	CAL Syntax	Description
040*ijkm*[4,5] or 041*ijkm*[4,5]	S*i* *exp*	Transmit *exp* into S*i* (040) or Transmit one's complement of *exp* into S*i* (041)
040*i*00*mn*[4,6] or 041*i*00*mn*[4,6]	S*i* *exp*	Transmit *exp* into S*i* (040) or Transmit one's complement of *exp* into S*i* (041)
042*i*00[2]	S*i* - 1	Enter -1 into S*i*
042*ijk*	S*i* <*exp*	Form one's mask in S*i* *of exp* bits from right; *jk* field gets 64-*exp*
042*ijk*[2]	S*i* #>*exp*	Form zeros mask in S*i* *of exp* bits from left; *jk* field gets 64-*exp*
042*i*77[2]	S*i* 1	Enter 1 into S*i*
043*i*00[2]	S*i* 0	Clear S*i*

Machine Instruction	CAL Syntax	Description
043*ijk*	S*i* >*exp*	Form one's mask in S*i of exp* bits from left; *jk* field gets *exp*
043*ijk*^{2}	S*i* #<*exp*	Form zeros mask in S*i of exp* bits from right; *jk* field gets 64-*exp*
047*i*00^2	S*i* #SB	Enter one's complement of sign bit into S*i*
051*i*00^2	S*i* SB	Enter sign bit into S*i*
071*i*30	S*i* 0.6	Transmit (0.75×2^{48}) as normalized floating-point constant into S*i*
071*i*40	S*i* 0.4	Transmit 0.5 as normalized floating-point constant into S*i*
071*i*50	S*i* 1.	Transmit 1.0 as normalized floating-point constant into S*i*
071*i*60	S*i* 2.	Transmit 2.0 as normalized floating-point constant into S*i*
071*i*70	S*i* 4.	Transmit 4.0 as normalized floating-point constant into S*i*

Transfers Into V Registers

The following instructions transmit values into the V registers.

Machine Instruction	CAL Syntax	Description
077*i*0*k*^{2}	V*i*,A*k* # 0	Clear element (A*k*) of register V*i*
145*iii*^{2}	V*i* 0	Clear V*i* elements

Transfers Into Semaphore Register

The following instructions transmit values into the Semaphore registers.

Machine Instruction	CAL Syntax	Description
0034*jk*	SM*jk* 1,TS	Test and set semaphore *jk*, $0<jk<31_{10}$
0036*jk*	SM*jk* 0	Clear semaphore *jk*, $0<jk<31_{10}$
0037*jk*	SM*jk* 1	Set semaphore *jk*, $0<jk<31_{10}$

Inter-register Transfer Instructions

The inter-register transfer instructions transmit the contents of one register to another register. In some cases, the register contents can be complemented, converted to floating-point format, or sign extended as a function of the transfer.

Transfers to A Registers

The following instructions transfer the contents of other registers into the A registers.

Machine Instruction	CAL Syntax	Description
023*ij*0	A*i* S*j*	Transmit (S*j*) to A*i*
023*i*01	A*i* VL	Transmit (VL) to A*i*
024*ijk*	A*i* B*jk*	Transmit (B*jk*) to A*i*
025*ijk*	B*jk* A*i*	Transmit (A*i*) to B*jk*
026*ij*7	A*i* SB*j*	Transmit (SB*j*) to A*i*
027*ij*7	SB*j* A*i*	Transmit (A*i*) to SB*j*
030*i*0*k*^{2}	A*i* A*k*	Transmit (A*k*) to A*i*
031*i*0*k*^{2}	A*i* -A*k*	Transmit negative of (A*k*) to A*i*
033*i*00	A*i* CI	Channel number of highest priority interrupt request to A*i* ($j=0$)
033*ij*0	A*i* CA,A*j*	Current address of channel (A*j*) to A*i* ($j \neq 0$, $k=0$)
033*ij*1	A*i* CE,A*j*	Error flag of channel (A*j*) to A*i* ($j \neq 0$, $k=1$)

Transfers to S Registers

The following instructions transmit the contents of other registers into the S registers.

Machine Instruction	CAL Syntax	Description
047*i*0*k*^{2}	S*i* #S*k*	Transmit one's complement of (S*k*) to S*i*
051*i*0*k*^{2}	S*i* S*k*	Transmit (S*k*) to S*i*
061*i*0*k*^{2}	S*i* -S*k*	Transmit negative of (S*k*) to S*i*
071*i*0*k*	S*i* A*k*	Transmit (A*k*) to S*i* with no sign extension

Machine Instruction	CAL Syntax	Description
071*i*1*k*	S*i* +A*k*	Transmit (A*k*) to S*i* with sign extension
071*i*2*k*	S*i* +FA*k*	Transmit (A*k*) to S*i* as unnormalized floating-point number
072*i*00	S*i* RT	Transmit (RTC) to S*i*
072*i*02	S*i* SM	Transmit semaphore to S*i*
072*ij*3	S*i* ST*j*	Transmit (ST*j*) register to S*i*
073*i*00	S*i* VM	Transmit (VM) to S*i*
073*i*01	S*i* SR	Transmit (SR*0*) to S*i* (*j*=0)
073*ij*3	ST*j* S*i*	Transmit (S*i*) to ST*j*
074*ijk*	S*i* T*jk*	Transmit (T*jk*) to S*i*
075*ijk*	T*jk* S*i*	Transmit (S*i*) to ST*jk*
076*ijk*	S*i* V*j*,A*k*	Transmit (V*j* element (A*k*)) to S*i*

Transfers to V Registers

The following instructions transmit the contents of other registers into the V registers.

Machine Instruction	CAL Syntax	Description
077*ijk*	V*i* ,A*k* S*j*	Transmit (S*j*) to V*i* element (A*k*)
142*i*0*k*^{2}	V*i* V*k*	Transmit (V*k* elements) to V*i* elements
156*i*0*k*^{2}	V*i* -V*k*	Transmit two's complement of (V*k* elements) to V*i* elements

Transfer to Vector Mask Register

The following instructions transmit the contents of other registers or zeroes into the Vector Mask register.

Machine Instruction	CAL Syntax	Description
0030*j*0	VM S*j*	Transmit (S*j*) to VM register
003000^2	VM 0	Clear VM register

Transfer to Vector Length Register

The following instructions transmit the contents of other registers into the Vector Length register.

Machine Instruction	CAL Syntax	Description
00200*k*	VL A*k*	Transmit (A*k*) to VL register
002000[2]	VL 1	Transmit 1 to VL register

Transfer to Semaphore Register

The following instruction transmits the contents of other registers into the Semaphore registers.

Machine Instruction	CAL Syntax	Description
073*i*02	SM S*i*	Load semaphores from S*i*

Memory Transfer Instructions

The memory transfer instructions enable/disable bidirectional memory transfers, transfer data between registers and memory, and ensure completion of memory references.

Bidirectional Memory Transfers

The following instructions enable or disable bidirectional memory transfers.

Machine Instruction	CAL Syntax	Description
002500	DBM	Disable bidirectional memory transfers
002600	EBM	Enable bidirectional memory transfers

Memory References

The following instruction ensures completion of instructions for bidirectional memory transfers.

Machine Instruction	CAL Syntax	Description
002700	CMR	Complete memory references

Stores

The following instructions store values into memory from operating registers.

Machine Instruction	CAL Syntax	Description
035*ijk*	,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
035*ijk*[2]	0,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
037*ijk*	,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
037*ijk*[2]	0,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
11*hijkm*[5]	*exp*,A*h* A*i*	Store (A*i*) to ((A*h*) + *exp*)
11*hi*00*mn*[6]	*exp*,A*h* A*i*	Store (A*i*) to ((A*h*) + *exp*)
11*hi*000[2,5]	,A*h* A*i*	Store (A*i*) to (A*h*)
11*hi*0000[2,6]	,A*h* A*i*	Store (A*i*) to (A*h*)
110*ijkm*[2,5]	*exp*,0 A*i*	Store (A*i*) to *exp*
110*i*00*mn*[2,6]	*exp*,0 A*i*	Store (A*i*) to *exp*
110*ijkm*[2,5]	*exp*, A*i*	Store (A*i*) to *exp*
110*i*00*mn*[2,6]	*exp*, A*i*	Store (A*i*) to *exp*
13*hijkm*[5]	*exp*,A*h* S*i*	Store (S*i*) to ((A*h*) + *exp*)
13*hi*00*mn*[6]	*exp*,A*h* S*i*	Store (S*i*) to ((A*h*) + *exp*)
130*ijkm*[2,5]	*exp*,0 S*i*	Store (S*i*) to *exp*
130*i*00*mn*[2,6]	*exp*,0 S*i*	Store (S*i*) to *exp*
130*ijkm*[2,5]	*exp*, S*i*	Store (S*i*) to *exp*
130*i*00*mn*[2,6]	*exp*, S*i*	Store (S*i*) to *exp*
13*hi*000[2,5]	,A*h* S*i*	Store (S*i*) to (A*h*)
13*hi*0000[2,6]	,A*h* S*i*	Store (S*i*) to (A*h*)
1770*jk*	,A0,A*k* V*j*	Store (V*j*) to memory starting at (A0) using (A*k*) as a constant increment

Machine Instruction	CAL Syntax	Description
1770*j*0	,A0,1 V *j*	Store (V*j*) to memory in consecutive addresses starting with (A0)
1771*jk*	,A0,V*k* V*j*	Store (V*j*) to memory using memory address (A0) + (V*k*)

Loads

The following instructions load values from memory into operating registers.

Machine Instruction	CAL Syntax	Description
034*ijk*	B*jk* ,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at address *jk*
034*ijk*[2]	B*jk*,A*i* 0,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at address *jk*
036*ijk*	T*jk*,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to T registers starting at address *jk*
036*ijk*[2]	T*jk*,A*i* 0,A0	Load (A*i*) words from memory starting at address (A0) to T registers starting at address *jk*
10*hijkm*[5]	A*i* *exp*,A*h*	Load from ((A*h*) + *exp*) to A*i*
10*hi*00*mn*[6]	A*i* *exp*,A*h*	Load from ((A*h*) + *exp*) to A*i*
10*hi*000[2,5]	A*i* ,A*h*	Load from (A*h*) to A*i*
10*hi*0000[2,6]	A*i* ,A*h*	Load from (A*h*) to A*i*
100*ijkm*[2,5]	A*i* *exp*,0	Load from (*exp*) to A*i*
100*i*00*mn*[2,6]	A*i* *exp*,0	Load from (*exp*) to A*i*
100*ijkm*[2,5]	A*i* *exp*,	Load from (*exp*) to A*i*
100*i*00*mn*[2,6]	A*i* *exp*,	Load from (*exp*) to A*i*
12*hijkm*[5]	S*i* *exp*,A*h*	Load from ((A*i*) + *exp*) to S*i*
12*hi*00*mn*[6]	S*i* *exp*,A*h*	Load from ((A*i*) + *exp*) to S*i*
120*ijkm*[2,5]	S*i* *exp*,0	Load from (*exp*) to S*i*
120*i*00*mn*[2,6]	S*i* *exp*,0	Load from (*exp*) to S*i*
120*ijkm*[2,5]	S*i* *exp*	Load from (*exp*) to S*i*

Machine Instruction	CAL Syntax	Description
120*i*00*mn*[2,6]	S*i* *exp*	Load from (*exp*) to S*i*
12*hi*000[2,5]	S*i* ,A*h*	Load from (A*h*) to S*i*
12*hi*0000[2,6]	S*i* ,A*h*	Load from (A*h*) to S*i*
176*i*0*k*	V*i* ,A0,A*k*	Load from memory starting at (A0) increased by (A*k*) and load into V*i*
176i00[2]	V*i* ,A0,1	Load from consecutive memory addresses starting with (A0) into V*i*
176i1k	V*i* ,A0,V*k*	Load from memory using memory address (A0) + (V*k*) into V*i*

Integer Arithmetic Instructions

Integer arithmetic operations obtain operands from registers and return results to registers. No direct memory references are possible.

The assembler recognizes several special syntax forms for increasing or decreasing register contents, such as the operands A*i*+1 and A*i*-1; however, these references actually result in register references such that the 1 becomes a reference to A*k* with *k*=0.

All integer arithmetic, whether 24-bit, 32-bit, or 64-bit, is two's complement and is represented as such in the registers. The Address Add and Address Multiply functional units perform 24-bit (X-mode) and 32-bit (Y-mode) arithmetic. The Scalar Add functional unit and the Vector Add functional unit perform 64-bit arithmetic. No overflow is detected by functional units when performing integer arithmetic.

Multiplication of two fractional operands is accomplished using a Floating-point Multiply instruction. The Floating-point Multiply functional unit recognizes conditions in which both operands have zero exponents as a special case and returns the high-order 48 bits of the result as an unnormalized fraction. Division of integers requires that they first be converted to floating-point format and then divided using the floating-point functional units. Refer to "Floating-point Arithmetic" earlier in this section for more information on these algorithms.

24-bit or 32-bit Integer Arithmetic

The following instructions perform 24-bit (X-mode) or 32-bit (Y-mode) integer arithmetic.

Machine Instruction	CAL Syntax	Description
030*ij*0[2]	A*i* A*j*+1	Integer sum of (A*j*) and 1 to A*i*

Machine Instruction	CAL Syntax	Description
030*ijk*	A*i* A*j*+A*k*	Integer sum of (A*j*) and (A*k*) to A*i*
031*ijk*	A*i* A*j*-A*k*	Integer difference of (A*j*) and (A*k*) to A*i*
031*ij*0^2	A*i* A*j*-1	Integer difference of (A*j*) and 1 to A*i*
032*ijk*	A*i* A*j**A*k*	Integer product of (A*j*) and (A*k*) to A*i*

64-bit Integer Arithmetic

The following instructions perform 64-bit integer arithmetic.

Machine Instruction	CAL Syntax	Description
060*ijk*	S*i* S*j*+S*k*	Integer sum of (S*j*) and (S*k*) to S*i*
061*ijk*	S*i* S*j*-S*k*	Integer difference of (S*j*) and (S*k*) to S*i*
154*ijk*	V*i* S*j*+V*k*	Integer sums of (S*j*) and (V*k* elements) to V*i* elements
155*ijk*	V*i* V*j*+V*k*	Integer sums of (V*j* elements) and (V*k* elements) to V*i* elements
156*ijk*	V*i* S*j*-V*k*	Integer differences of (S*j*) and (V*k* elements) to V*i* elements
157*ijk*	V*i* V*j*-V*k*	Integer differences of (V*j* elements) and (V*k* elements) to V*i* elements

Floating-point Arithmetic Instructions

All floating-point arithmetic operations use registers as the source of operands and return results to registers.

Floating-point numbers are represented in a standard format throughout the CPU. This format is a packed representation of a binary coefficient and an exponent or power of 2. The coefficient is a 48-bit signed fraction. The sign of the coefficient is separated from the rest of the coefficient. Because the coefficient is signed magnitude, it is not complemented for negative values. Refer to "Floating-point Arithmetic" earlier in this section for more information on floating-point numbers and arithmetic.

Floating-point Range Errors

The following instructions enable or disable Floating-point Range errors to be flagged.

Machine Instruction	CAL Syntax	Description
002100	EFI	Enable interrupt on Floating-point error
002200	DFI	Disable interrupt on Floating-point error

Floating-point Addition and Subtraction

The following instructions perform floating-point addition or subtraction.

Machine Instruction	CAL Syntax	Description
062*ijk*	S*i* S*j*+FS*k*	Floating-point sum of (S*j*) and (S*k*) to S*i*
062*i*0*k*[2]	S*i* +FS*k*	Normalize (S*k*) to S*i*
063*ijk*	S*i* S*j*-FS*k*	Floating-point difference of (S*j*) and (S*k*) to S*i*
063*i*0*k*[2]	S*i* -FS*k*	Transmit the normalized negative of (S*k*) to S*i*
170*ijk*	V*i* S*j*+FV*k*	Floating-point sums of (S*j*) and (V*k* elements) to V*i* elements
170*i*0*k*[2]	V*i* + FV*k*	Transmit normalized (V*k* elements) to V*i* elements
171*ijk*	V*i* V*j*+FV*k*	Floating-point sums of (V*j* elements) and (V*k* elements) to V*i* elements
172*ijk*	V*i* S*j*-FV*k*	Floating-point differences of (S*j*) and (V*k* elements) to V*i* elements
172*i*0*k*[2]	V*i* -FV*k*	Transmit normalized negative of (V*k* elements) to V*i* elements
173*ijk*	V*i* V*j*-FV*k*	Floating-point differences of (V*j* elements) and (V*k* elements) to V*i* elements

Floating-point Multiplication

The following instructions perform floating-point multiplication.

Machine Instruction	CAL Syntax	Description
064*ijk*	S*i* S*j**FS*k*	Floating-point product of (S*j*) and (S*k*) to S*i*
065*ijk*	S*i* S*j**HS*k*	Half-precision rounded floating-point product of (S*j*) and (S*k*) to S*i*
066*ijk*	S*i* S*j**RS*k*	Rounded floating-point product of (S*j*) and (S*k*) to S*i*
160*ijk*	V*i* S*j**FV*k*	Floating-point products of (S*j*) and (V*k* elements) to V*i* elements
161*ijk*	V*i* V*j**FV*k*	Floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements
162*ijk*	V*i* S*j**HV*k*	Half-precision rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements
163*ijk*	V*i* V*j**HV*k*	Half-precision rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements
164*ijk*	V*i* S*j**RV*k*	Rounded floating-point products of (S*j*) and (V*k* elements) to V*i* elements
165*ijk*	V*i* V*j**RV*k*	Rounded floating-point products of (V*j* elements) and (V*k* elements) to V*i* elements

Reciprocal Iteration

The following instructions perform reciprocal iteration operations.

Machine Instruction	CAL Syntax	Description
067*ijk*	S*i* S*j**IS*k*	Reciprocal iteration 2 - (S*j*)×(S*k*) to S*i*
166*ijk*[5]	V*i* S*j**IV*k*	Reciprocal iteration 2 - (S*j*)×(V*k* elements) to V*i* elements
166*ijk*[6]	V*i* S*j**V*k*	32-bit integer product of (S*j*) and (V*k* elements) to V*i* elements
167*ijk*	V*i* V*j**IV*k*	Reciprocal iteration 2 - (V*j* elements) × (V*k* elements) to V*i* elements

Reciprocal Approximation

The following instructions perform floating-point reciprocal approximation operations.

Machine Instruction	CAL Syntax	Description
070*ij*0	S*i* /HS*j*	Floating-point reciprocal approximation of (S*j*) to S*i*
174*ij*0	V*i* /HV*j*	Floating-point reciprocal approximation of (V*j* elements) to V*i* elements

Logical Operation Instructions

The Scalar and Vector Logical functional units perform bit-by-bit manipulation of 64-bit quantities. Logical operations include logical products, logical sums, logical differences, logical equivalence, Vector Mask, and merges. Logical operations are defined below.

- A logical product (& operator) is the AND function.
- A logical difference (\ operator) is the EXCLUSIVE OR function.
- A logical sum (! operator) is the INCLUSIVE OR function.
- A logical merge combines two operands depending on a one's mask in a third operand. The result is defined by (operand 2 & mask) ! (operand 1 & #mask).

Logical Products

The following instructions produce logical products.

Machine Instruction	CAL Syntax	Description
044*ijk*	S*i* S*j*&S*k*	Logical product of (S*j*) and (S*k*) to S*i*
044*ij*0^2	S*i* S*j*&SB	Sign bit of (S*j*) to S*i*
044*ij*0^2	S*i* SB&S*j*	Sign bit of (S*j*) to S*i* ($j \neq 0$)
045*ijk*	S*i* #S*k*&S*j*	Logical product of (S*j*) and complement of (S*k*) to S*i*
045*ij*0^2	S*i* #SB&S*j*	(S*j*) with sign bit cleared to S*i*
140*ijk*	V*i* S*j*&V*k*	Logical products of (S*j*) and (V*k* elements) to V*i* elements
141*ijk*	V*i* V*j*&V*k*	Logical products of (V*j* elements) and (V*k* elements) to V*i* elements

Logical Sums

The following instructions produce logical sums.

Machine Instruction	CAL Syntax	Description
051*ijk*	S*i* S*j*!S*k*	Logical sum of (S*j*) and (S*k*) to S*i*
051*ij*0[2]	S*i* S*j*!SB	Logical sum of (S*j*) and sign bit to S*i*
051*ij*0[2]	S*i* SB!S*j*	Logical sum of (S*j*) and sign bit to S*i* (j ≠ 0)
142*ijk*	V*i* S*j*!V*k*	Logical sums of (S*j*) and (V*k* elements) to V*i* elements
143*ijk*	V*i* V*j*!V*k*	Logical sums of (V*j* elements) and (V*k* elements) to V*i* elements

Logical Differences

The following instructions produce logical differences.

Machine Instruction	CAL Syntax	Description
046*ijk*	S*i* S*j*\S*k*	Logical difference of (S*j*) and (S*k*) to S*i*
046*ij*0[2]	S*i* S*j*\SB	Toggle sign bit of (S*j*), then enter into S*i*
046*ij*0[2]	S*i* SB\S*j*	Toggle sign bit of (S*j*), then enter into (S*i*) (*j* ≠ 0)
144*ijk*	V*i* S*j*\V*k*	Logical differences of (S*j*) and (V*k* elements) to V*i* elements
145*ijk*	V*i* V*j*\V*k*	Logical differences of (V*j* elements) and (V*k* elements) to V*i* elements

Logical Equivalence

The following instructions produce logical equivalence.

Machine Instruction	CAL Syntax	Description
047*ijk*	S*i* #S*j*\S*k*	Logical equivalence of (S*j*) and (S*k*) to S*i*
047*ij*0[2]	S*i* #S*j*\SB	Logical equivalence of (S*j*) and sign bit to S*i*
047*ij*0[2]	S*i* #SB\S*j*	Logical equivalence of (S*j*) and sign bit to S*i* (*j* ≠ 0)

Vector Mask

The following instructions perform a mask operation that sets a vector operand for certain elements depending on the mask.

Machine Instruction	CAL Syntax	Description
1750*j*0	VM V*j*,Z	Set VM bits for zero elements of V*j*
1750*j*1	VM V*j*,N	Set VM bits for nonzero elements of V*j*
1750*j*2	VM V*j*,P	Set VM bits for positive elements of V*j*
1750*j*3	VM V*j*,M	Set VM bits for negative elements of V*j*
175*ij*4	V*i*,VM V*j*,Z	Set VM bits and register V*i* to V*j*, for zero elements of V*j*
175*ij*5	V*i*,VM V*j*,N	Set VM bits and register V*i* to V*j*, for nonzero elements of V*j*
175*ij*6	V*i*,VM V*j*,P	Set VM bits and register V*i* to V*j*, for positive elements of V*j*
175*ij*7	V*i*,VM V*j*,M	Set VM bits and register V*i* to V*j*, for negative elements of V*j*

Merge

The following instructions perform a logical merge that combines two operands depending on a one's mask in a third operand.

Machine Instruction	CAL Syntax	Description
050*ijk*	S*i* S*j*!S*i*&S*k*	Logical product of (S*i*) and (S*k*) complemented ORed with logical product of (S*j*) and (S*k*) to S*i*
050*ij*0[2]	S*i* S*j*!S*i*&SB	Scalar merge of (S*i*) and sign bit of (S*j*) to S*i*
146*ijk*	V*i* S*j*!V*k*&VM	Transmit (S*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*
146*i*0*k*[2]	V*i* #VM&V*k*	Vector merge of (V*k*) and 0 to V*i*
147*ijk*	V*i* V*j*!V*k*&VM	Transmit (V*j*) if VM bit=1; (V*k*) if VM bit=0 to V*i*

Shift Instructions

The Scalar Shift functional unit and Vector Shift functional unit shift 64-bit quantities or 128-bit quantities. A 128-bit quantity is formed by concatenating two 64-bit quantities. The number of bits a value is shifted left or right is determined by the value of an expression for some instructions and by the contents of an A register for other instructions. If the count is specified by an expression, the value of the expression must not exceed 64.

Machine Instruction	CAL Syntax	Description
052*ijk*	S0 S*i*<*exp*	Shift (S*i*) left *exp* places to S0; *exp*=*jk*
053*ijk*	S0 S*i*>*exp*	Shift (S*i*) right *exp* places to S0; *exp*=64-*jk*
054*ijk*	S*i* S*i*<*exp*	Shift (S*i*) left *exp* places to S*i*; *exp*=*jk*
055*ijk*	S*i* S*i*>*exp*	Shift (S*i*) right *exp* places to S*i*; *exp*=64-*exp*
056*ijk*	S*i* S*i*,S*j*<A*k*	Shift (S*i*) and (S*j*) left by (A*k*) places to S*i*
056*ij*0[2]	S*i* S*i*,S*j*<1	Shift (S*i*) and (S*j*) left one place to S*i*
056*i*0*k*[2]	S*i* S*i*<A*k*	Shift (S*i*) left (A*k*) places to S*i*
057*ijk*	S*i* S*j*,S*i*>A*k*	Shift (S*j*) and (S*i*) right by (A*k*) places to (S*i*)
057*ij*0[2]	S*i* S*j*,S*i*>1	Shift (S*j*) and (S*i*) right one place to (S*i*)
057*i*0*k*[2]	S*i* S*i*>A*k*	Shift (S*i*) right (A*k*) places to S*i*
150*ijk*	V*i* V*j*<A*k*	Shift (V*j* elements) left by (A*k*) places to V*i* elements
150*ij*0[2]	V*i* V*j*<1	Shift (V*j* elements) left one place to V*i* elements
151*ijk*	V*i* V*j*>A*k*	Shift (V*j* elements) right by (A*k*) places to V*i* elements
151*ij*0[2]	V*i* V*j*>1	Shift (V*j* elements) right one place to V*i* elements
152*ijk*	V*i* V*j*,V*j*<A*k*	Double shift of (V*j* elements) left (A*k*) places to V*i* elements
152*ij*0[2]	V*i* V*j*,V*j*<1	Double shift of (V*j* elements) left one place to V*i* elements
153*ijk*	V*i* V*j*,V*j*>A*k*	Double shift of (V*j* elements) right (A*k*) places to V*i* elements
153*ij*0[2]	V*i* V*j*,V*j*>1	Double shift of (V*j* elements) right one place to V*i* elements

Bit Count Instructions

Bit count instructions count the number of set bits or the number of leading 0 bits in an S or V register.

Scalar Population Count

The following instruction performs the scalar population count.

Machine Instruction	CAL Syntax	Description
026*ij*0	A*i* PS*j*	Population count of (S*j*) to A*i*

Vector Population Count

The following instruction performs the vector population count.

Machine Instruction	CAL Syntax	Description
174*ij*1	V*i* PV*j*	Population count of (V*j* elements) to (V*i* elements)

Population Count Parity

The following instructions perform population parity count.

Machine Instruction	CAL Syntax	Description
026*ij*1	A*i* QS*j*	Population parity count of (S*j*) to A*i*
174*ij*2	V*i* QV*j*	Population parity count of (V*j* elements) to (V*i* elements)

Scalar Leading Zero Count

The following instruction performs leading zero count.

Machine Instruction	CAL Syntax	Description
027*ij*0	A*i* ZS*j*	Leading zero count of (S*j*) to A*i*

Branch Instructions

Instructions in this category include conditional and unconditional branch instructions. An expression or the contents of a B register specify the branch address. An address is always taken to be a parcel address when the instruction runs. If an expression has a word-address attribute, the assembler issues an error message.

Unconditional Branch Instructions

The following instructions perform unconditional branch operations.

Machine Instruction	CAL Syntax	Description
0050*jk*	J B*jk*	Jump to (B*jk*)
006*ijk*m	J *exp*	Jump to *exp*

Conditional Branch Instructions

The following instructions perform conditional branch operations.

Machine Instruction	CAL Syntax	Description
010*ijkm*	JAZ *exp*	Jump to *exp* if (A0) = 0 ($i2 = 0$)
011*ijkm*	JAN *exp*	Jump to *exp* if (A0) ≠ 0 ($i2 = 0$)
012*ijkm*	JAP *exp*	Jump to *exp* if (A0) positive; includes (A0) = 0 ($i2 = 0$)
013*ijkm*	JAM *exp*	Jump to *exp* if (A0) negative ($i2 = 0$)
014*ijkm*	JSZ *exp*	Jump to *exp* if (S0) = 0 ($i2 = 0$)
015*ijkm*	JSN *exp*	Jump to *exp* if (S0) ≠ 0 ($i2 = 0$)
016*ijkm*	JSP *exp*	Jump to *exp* if (S0) positive; includes (S0) = 0 ($i2 = 0$)
017*ijkm*	JSM *exp*	Jump to *exp* if (S0) negative ($i2 = 0$)

Return Jump

The following instructions perform return jump operations.

Machine Instruction	CAL Syntax	Description
001000[2]	PASS	Pass or no-operation
007*ijkm*	R *exp*	Return jump to *exp*; set B00 to (P) + 2

Normal Exit

The following instruction performs normal exit operations.

Machine Instruction	CAL Syntax	Description
004000	EX	Normal exit

Error Exit

The following instruction performs error exit operations.

Machine Instruction	CAL Syntax	Description
000000	ERR	Error exit

Monitor Mode Instructions

Monitor mode instructions are executed only when the CPU is in monitor mode. An attempt to execute one of these instructions when not in monitor mode is treated as a Pass instruction. The instructions perform specialized functions useful to the operating system.

Channel Control

The following instructions perform channel control operations.

Machine Instruction	CAL Syntax	Description
0010*jk*[1]	CA,A*j* A*k*	Set the CA register for the channel indicated by (A*j*) to (A*k*) and activate the channel
0011*jk*[1]	CL,A*j* A*k*	Set the CL register for the channel indicated by (A*j*) to (A*k*) address

Machine Instruction	CAL Syntax	Description
0012*j*0[1]	CI,A*j*	Clear the Interrupt flag and Error flag for the channel indicated by (A*j*); clear device master-clear (output channel)
0012*j*1[1]	MC,A*j*	Clear the Interrupt flag and Error flag for the channel indicated by (A*j*); set device master-clear (output channel); clear device ready-held (input channel)
0013*j*0[1]	XA A*j*	Enter XA register with (A*j*)

Set Real-time Clock

The following instruction performs Real-time Clock operations.

Machine Instruction	CAL Syntax	Description
0014*j*0[1]	RT S*j*	Load RTC register with (S*j*)

Programmable Clock Interrupt Instructions

The following instructions perform Programmable Clock operations.

Machine Instruction	CAL Syntax	Description
0014*j*4[1]	PCI S*j*	Load II register with (S*j*)
001405[1]	CCI	Clear Programmable Clock Interrupt request
001406[1]	ECI	Enable Programmable Clock Interrupt request
001407[1]	DCI	Disable Programmable Clock Interrupt request

Interprocessor Interrupt Instructions

The following instructions perform Interprocessor Interrupt operations.

Machine Instruction	CAL Syntax	Description
0014*j*1[1]	SIPI A*j*	Set Interprocessor Interrupt request to CPU (A*j*)
001401[1,2]	SIPI	Set Interprocessor Interrupt request to CPU 0
001402[1]	CIPI	Clear Interprocessor Interrupt

Cluster Number Instructions

The following instruction sets the cluster number.

Machine Instruction	CAL Syntax	Description
0014*j*3[1]	CLN A*j*	Load CLN register with (A*j*) where $0 \leq exp \leq 9$

Operand Range Error Interrupt Instructions

The following instructions enable or disable Operand Range Error interrupts.

Machine Instruction	CAL Syntax	Description
002300[1]	ERI	Enable interrupt on Address Range error
002400[1]	DRI	Disable interrupt on Address Range error

Performance Counters

The following instructions select performance monitor features.

Machine Instruction	CAL Syntax	Description
0015*j*0[1,3]		Select performance monitor
073*i*11[1,3]		Read performance counter into S*i*
073021[1,3]		Increment performance counter

Maintenance Mode

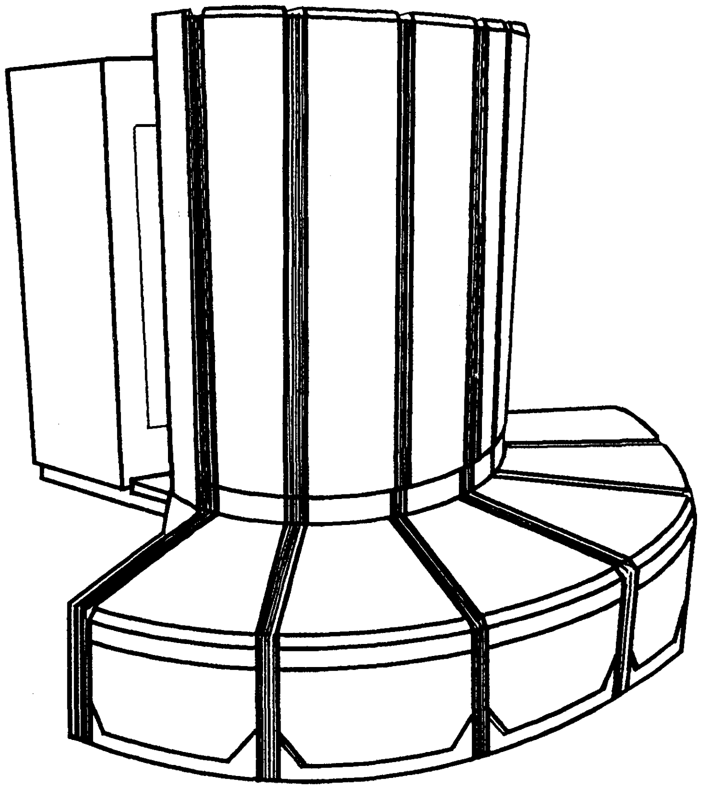
The following instructions select maintenance features.

Machine Instruction	CAL Syntax	Description
001501[1,3]		Set maintenance read mode
001511[1,3]		Load diagnostic check byte with S1
001521[1,3]		Set maintenance write mode 1
001531[1,3]		Set maintenance write mode 2
073031[1,3]		Clear all maintenance modes

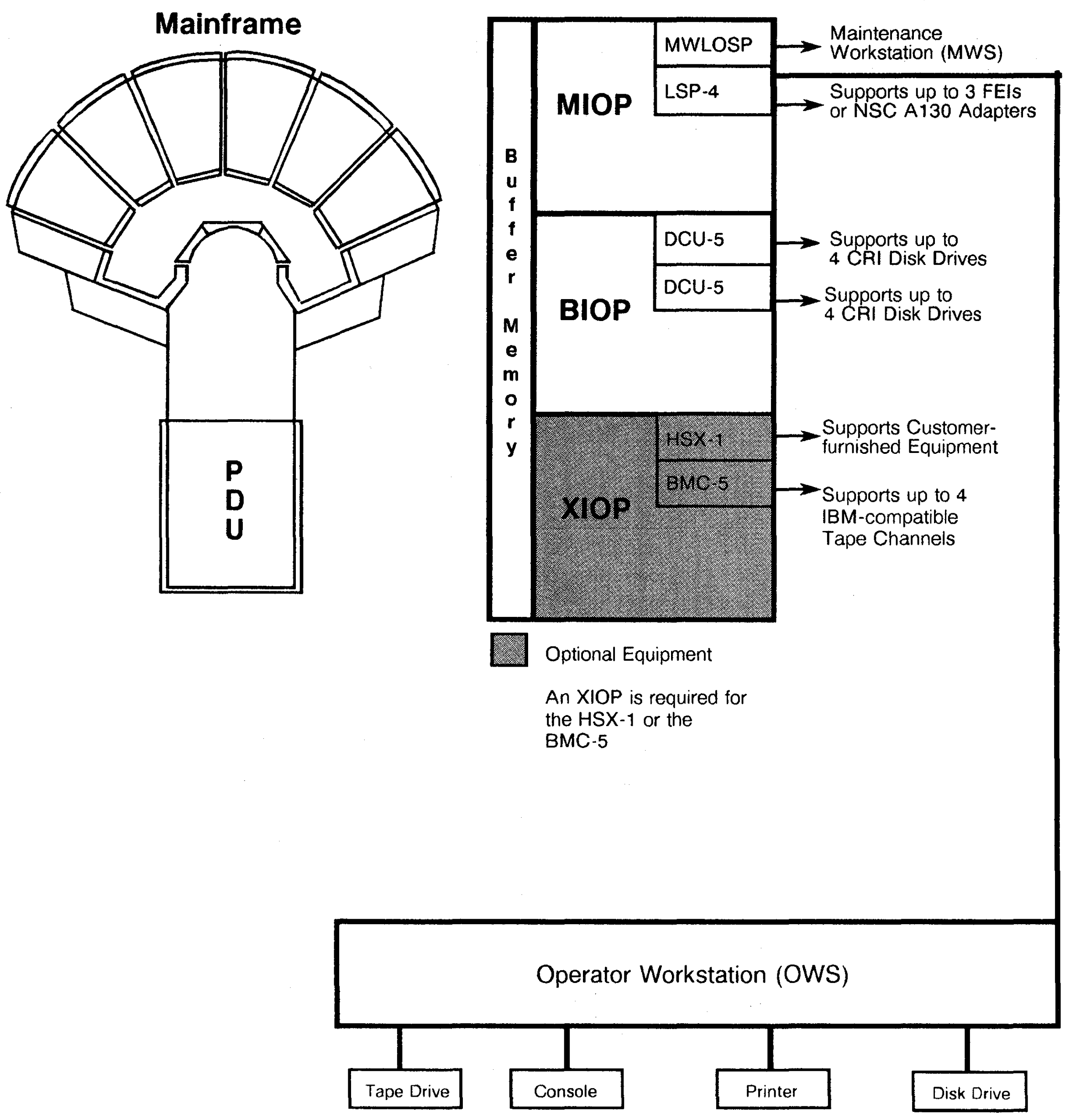
Note: The following footnotes are used throughout the instruction summary:

Footnote	Description
1	Privileged to monitor mode
2	Special syntax mode
3	Not supported by CAL Version 2
4	Generated depending on the value of *exp*
5	X-mode only
6	Y-mode only

CRAY X-MP EA/1se COMPUTER SYSTEM



Specification Sheet
HR-3020 EA/1se


Mainframe
PDU
Buffer Memory
MIOP
MWLOSP
LSP-4
Maintenance Workstation (MWS)
Supports up to 3 FEIs or NSC A130 Adapters
BIOP
DCU-5
DCU-5
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
BMC-5
Supports Customer-furnished Equipment
Supports up to 4 IBM-compatible Tape Channels
▨ Optional Equipment
An XIOP is required for the HSX-1 or the BMC-5
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

CRAY X-MP EA/1se Computer System Maximum Configuration

CRAY X-MP EA/1se MAINFRAME FEATURES

System Clock

Speed	10.0 ns

CPU Specifications

Number of CPUs	1

Number of Registers per CPU

Address Registers (A) 32 bits each	8
Intermediate Registers (B) 32 bits each	64
Scalar Registers (S) 64 bits each	8
Scalar-save Registers (T) 64 bits each	64
Vector Registers (V) 64 bits per element 64 elements per register	8

Number of Functional Units per CPU

Address Addition	1
Address Multiplication	1
Scalar Addition	1
Scalar Shift	1
Scalar Logical	1
Scalar Population Parity/Leading Zero	1
Vector Addition	1
Vector Multiplication	1
Full Vector Logical	1
2nd Vector Logical	1
Vector Population/Parity	1
Floating-point Addition	1
Floating-point Multiplication	1
Floating-point Reciprocal	1

Shared Resources

Central Memory

Word Width	64 bits
SECDED Error Correction	8 bits
Memory Size	4, 16 Mwords
Number of Banks	16
Number of Ports per CPU	4

I/O Section

100 Mbyte/s Channels	2
6 Mbyte/s Channels	4

Shared Registers, Three Clusters Consisting of:

Shared Address (SB) 32 bits each (Y-mode) 24 bits each (X-mode)	8
Shared Scalar (ST) 64 bits each	8
Semaphore (SM) 1 bit each	32

Real-time Clock (64 bits)	1

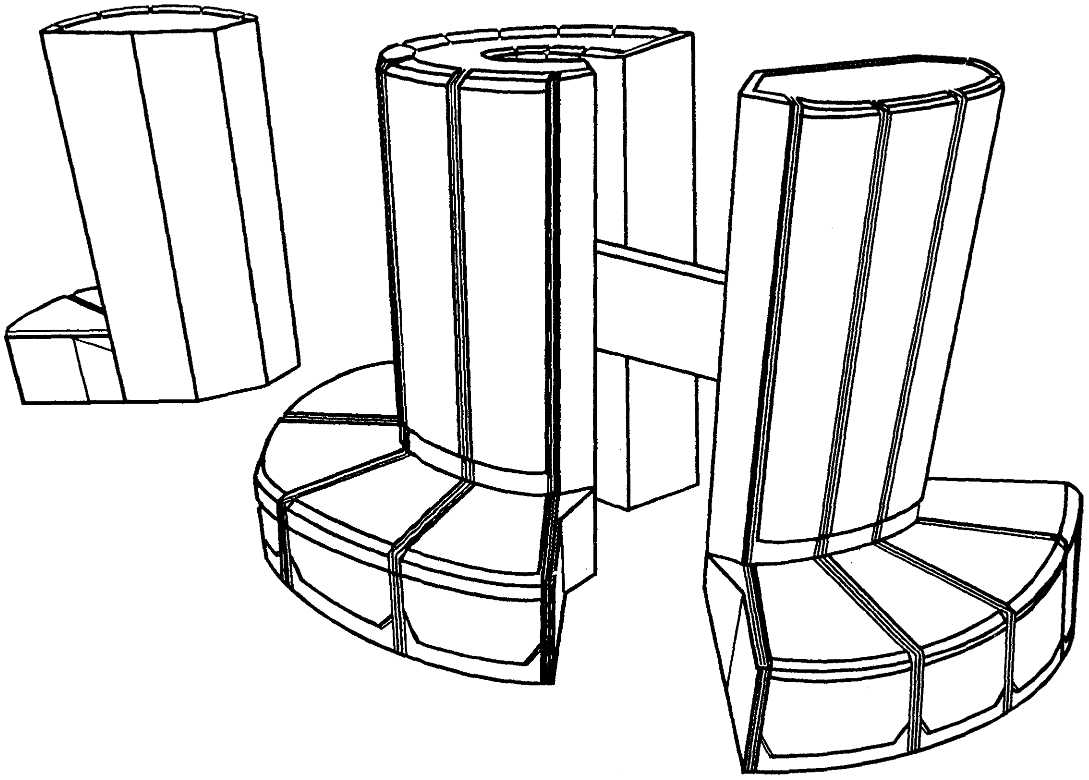
PHYSICAL DESCRIPTION

Floor Space	21.06 ft^2 (1.8 m^2)
Weight	5,720 lbs (2,595 kg)
Columns	6

SUPPORT EQUIPMENT

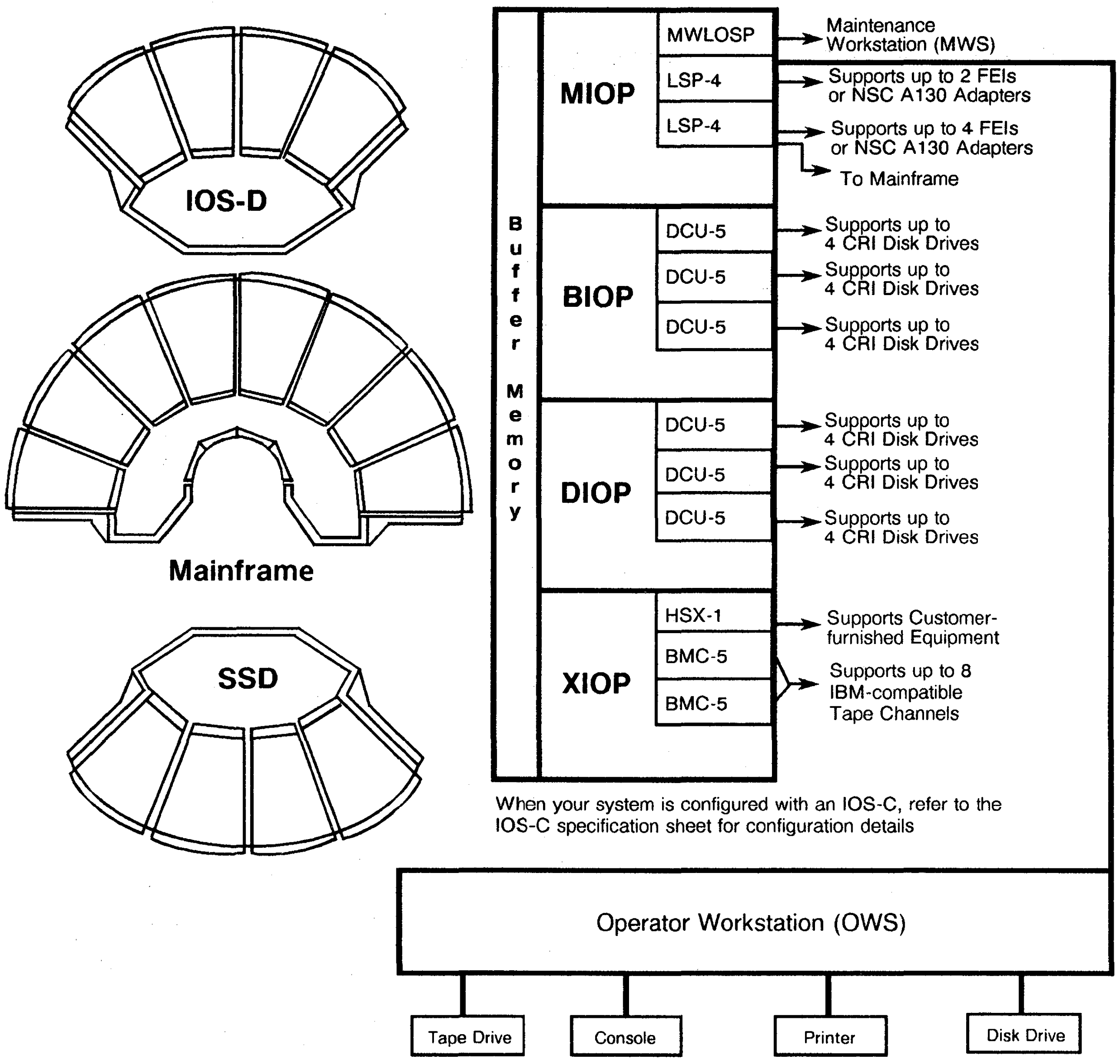
Refrigeration Condensing Unit	1
Motor Generator Sets	1
Operator Workstation	1
Maintenance Workstation	1
Power Distribution Unit	1

CRAY X-MP EA/1 COMPUTER SYSTEM



Specification Sheet

HR-3020/1


IOS-D
Mainframe
SSD
Buffer Memory
MIOP
MWLOSP
LSP-4
LSP-4
Maintenance Workstation (MWS)
Supports up to 2 FEIs or NSC A130 Adapters
Supports up to 4 FEIs or NSC A130 Adapters
To Mainframe
BIOP
DCU-5
DCU-5
DCU-5
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
DIOP
DCU-5
DCU-5
DCU-5
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
BMC-5
BMC-5
Supports Customer-furnished Equipment
Supports up to 8 IBM-compatible Tape Channels
When your system is configured with an IOS-C, refer to the IOS-C specification sheet for configuration details
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

CRAY X-MP EA/1 Computer System Maximum Configuration

CRAY X-MP EA/1 MAINFRAME FEATURES

System Clock

Speed	8.5 ns

CPU Specifications

Number of CPUs	1

Number of Registers per CPU

Address Registers (A) 32 bits each	8
Intermediate Registers (B) 32 bits each	64
Scalar Registers (S) 64 bits each	8
Scalar-save Registers (T) 64 bits each	64
Vector Registers (V) 64 bits per element 64 elements per register	8

Number of Functional Units per CPU

Address Addition	1
Address Multiplication	1
Scalar Addition	1
Scalar Shift	1
Scalar Logical	1
Scalar Population Parity/Leading Zero	1
Vector Addition	1
Vector Multiplication	1
Full Vector Logical	1
2nd Vector Logical	1
Vector Population/Parity	1
Floating-point Addition	1
Floating-point Multiplication	1
Floating-point Reciprocal	1

Shared Resources

Central Memory

Word Width	64 bits
SECDED Error Correction	8 bits
Memory Size	16, 32, 64 Mwords
Number of Banks	32, 64
Number of Ports per CPU	4

I/O Section

1,000 Mbyte/s Channels	1
100 Mbyte/s Channels	2
6 Mbyte/s Channels	4

Shared Registers, Three Clusters Consisting of:

Shared Address (SB) 32 bits each (Y-mode) 24 bits each (X-mode)	8
Shared Scalar (ST) 64 bits each	8
Semaphore (SM) 1 bit each	32

Real-time Clock (64 bits)	1

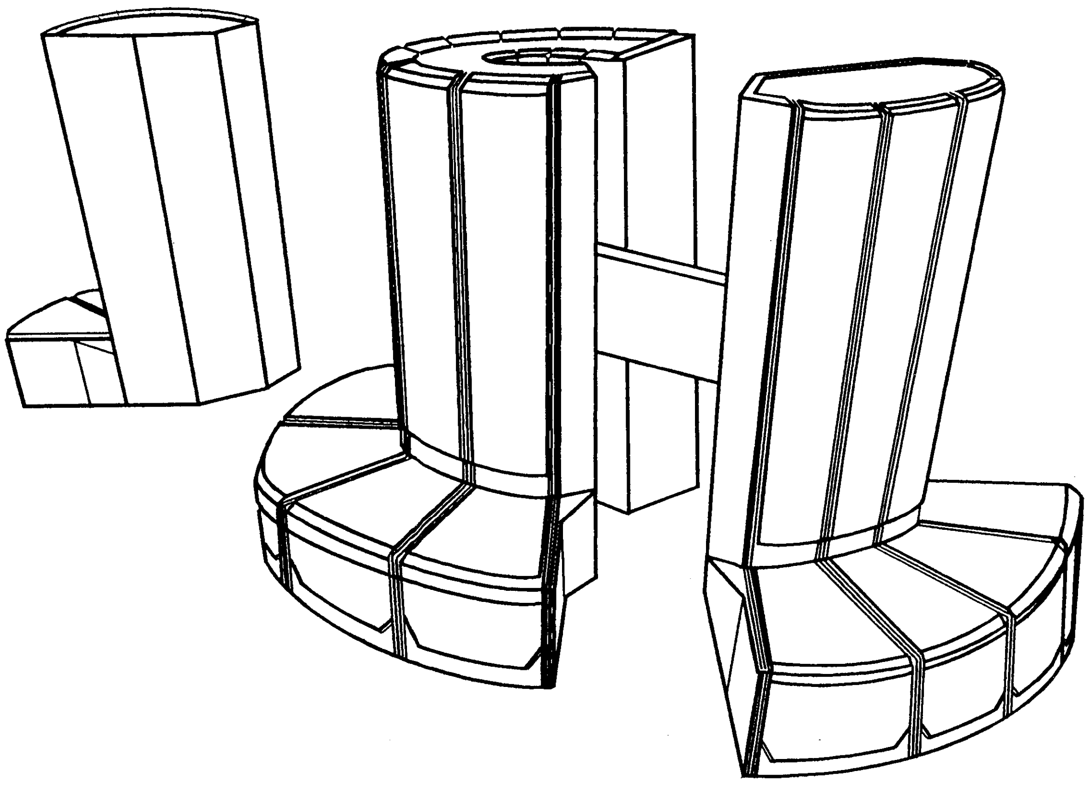
PHYSICAL DESCRIPTION

Floor Space	28.38 ft^2 (2.39 m^2)
Weight	7,560 lbs (3,429 kgs)
Columns	8

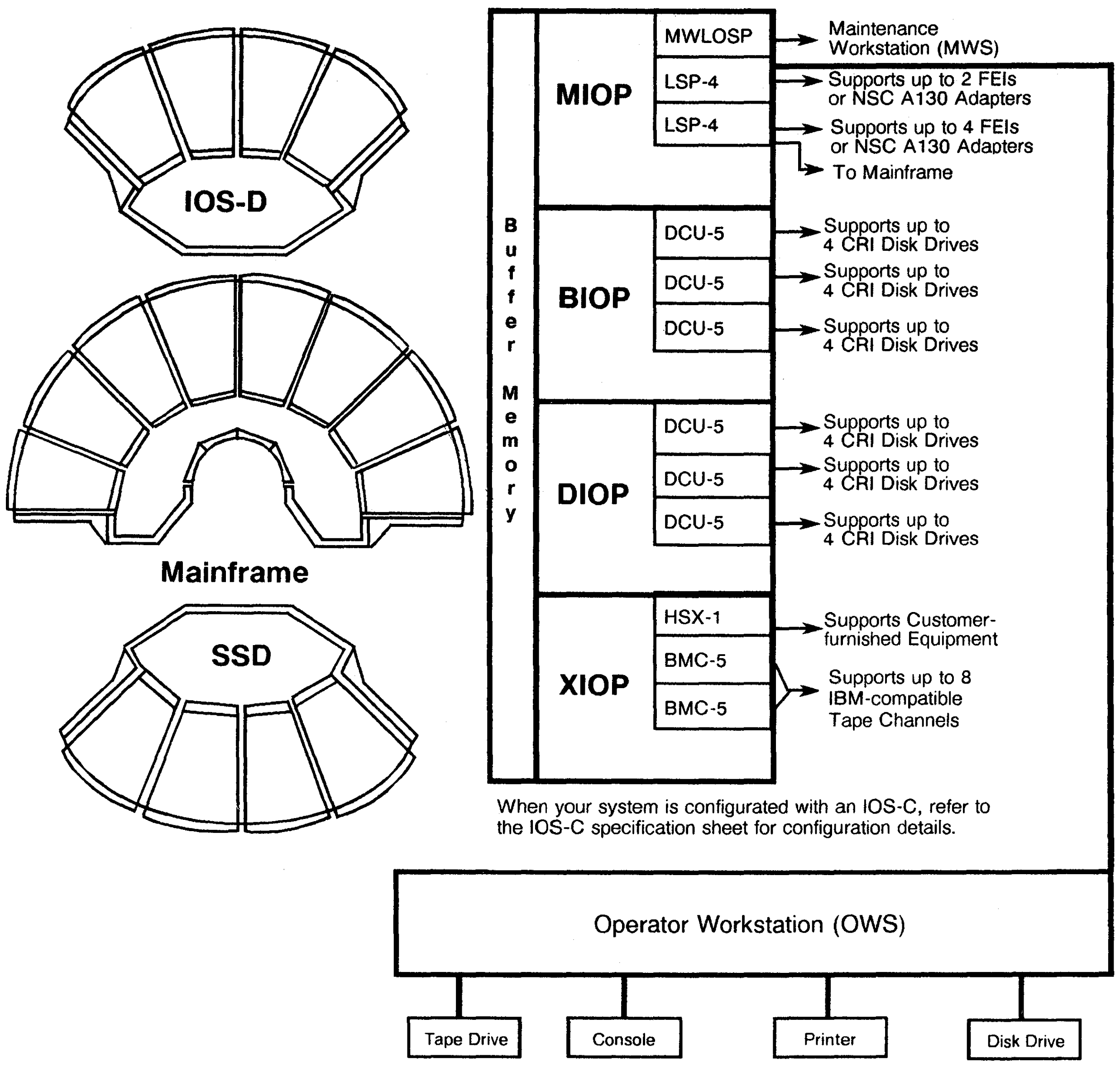
SUPPORT EQUIPMENT

Refrigeration Condensing Unit	1
Motor Generator Sets	2
Operator Workstation	1
Maintenance Workstation	1
Power Distribution Unit	2 or 3

CRAY X-MP EA/2 COMPUTER SYSTEM



Specification Sheet
HR-3020/2


IOS-D
Mainframe
SSD
Buffer Memory
MIOP
MWLOSP
Maintenance Workstation (MWS)
LSP-4
Supports up to 2 FEIs or NSC A130 Adapters
LSP-4
Supports up to 4 FEIs or NSC A130 Adapters
To Mainframe
BIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
Supports Customer-furnished Equipment
BMC-5
BMC-5
Supports up to 8 IBM-compatible Tape Channels
When your system is configurated with an IOS-C, refer to the IOS-C specification sheet for configuration details.
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

CRAY X-MP EA/2 Computer System Maximum Configuration

CRAY X-MP EA/2 MAINFRAME FEATURES

System Clock

Speed	8.5 ns

CPU Specifications

Number of CPUs	2

Number of Registers per CPU

Address Registers (A) 32 bits each	8
Intermediate Registers (B) 32 bits each	64
Scalar Registers (S) 64 bits each	8
Scalar-save Registers (T) 64 bits each	64
Vector Registers (V) 64 bits per element 64 elements per register	8

Number of Functional Units per CPU

Address Addition	1
Address Multiplication	1
Scalar Addition	1
Scalar Shift	1
Scalar Logical	1
Scalar Population Parity/Leading Zero	1
Vector Addition	1
Vector Multiplication	1
Full Vector Logical	1
2nd Vector Logical	1
Vector Population/Parity	1
Floating-point Addition	1
Floating-point Multiplication	1
Floating-point Reciprocal	1

Shared Resources

Central Memory

Word Width	64 bits
SECDED Error Correction	8 bits
Memory Size	16, 32, 64 Mwords
Number of Banks	32, 64
Number of Ports per CPU	4

I/O Section

1,000 Mbyte/s Channels	1
100 Mbyte/s Channels	2
6 Mbyte/s Channels	4

Shared Registers, Three Clusters Consisting of:

Shared Address (SB) 32 bits each (Y-mode) 24 bits each (X-mode)	8
Shared Scalar (ST) 64 bits each	8
Semaphore (SM) 1 bit each	32

Real-time Clock (64 bits)	1

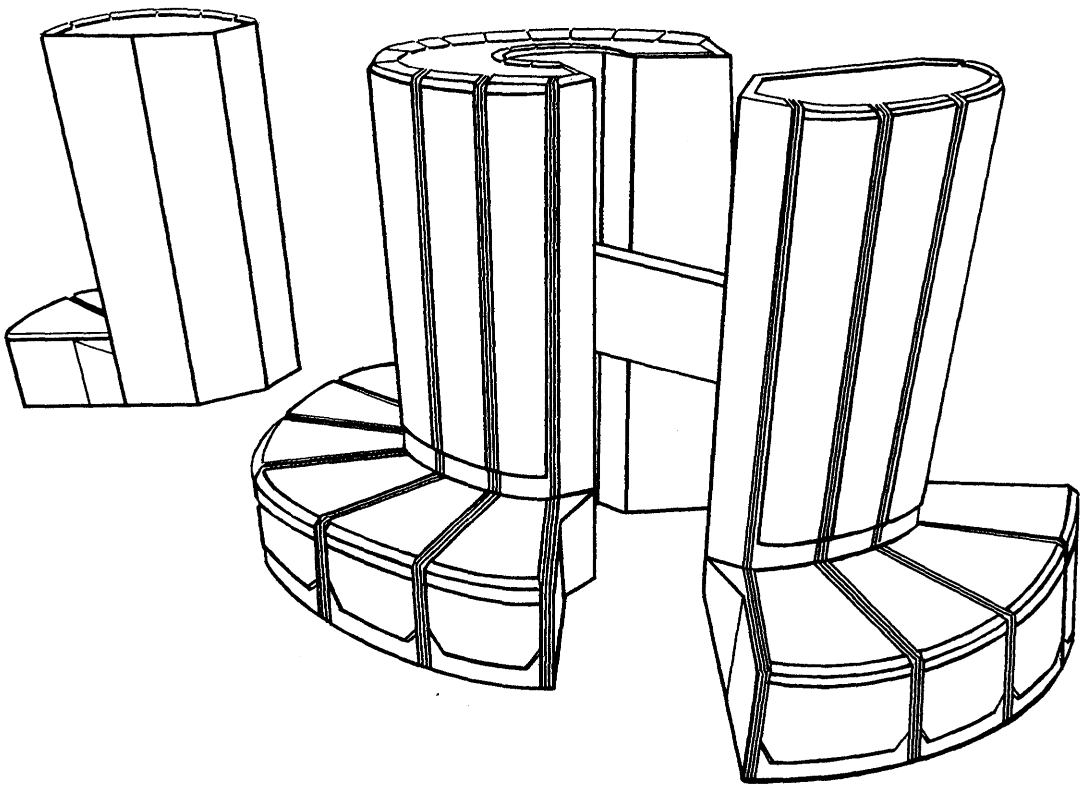
PHYSICAL DESCRIPTION

Floor Space	28.38 ft^2 (2.39 m^2)
Weight	7,560 lbs (3,429 kgs)
Columns	8

SUPPORT EQUIPMENT

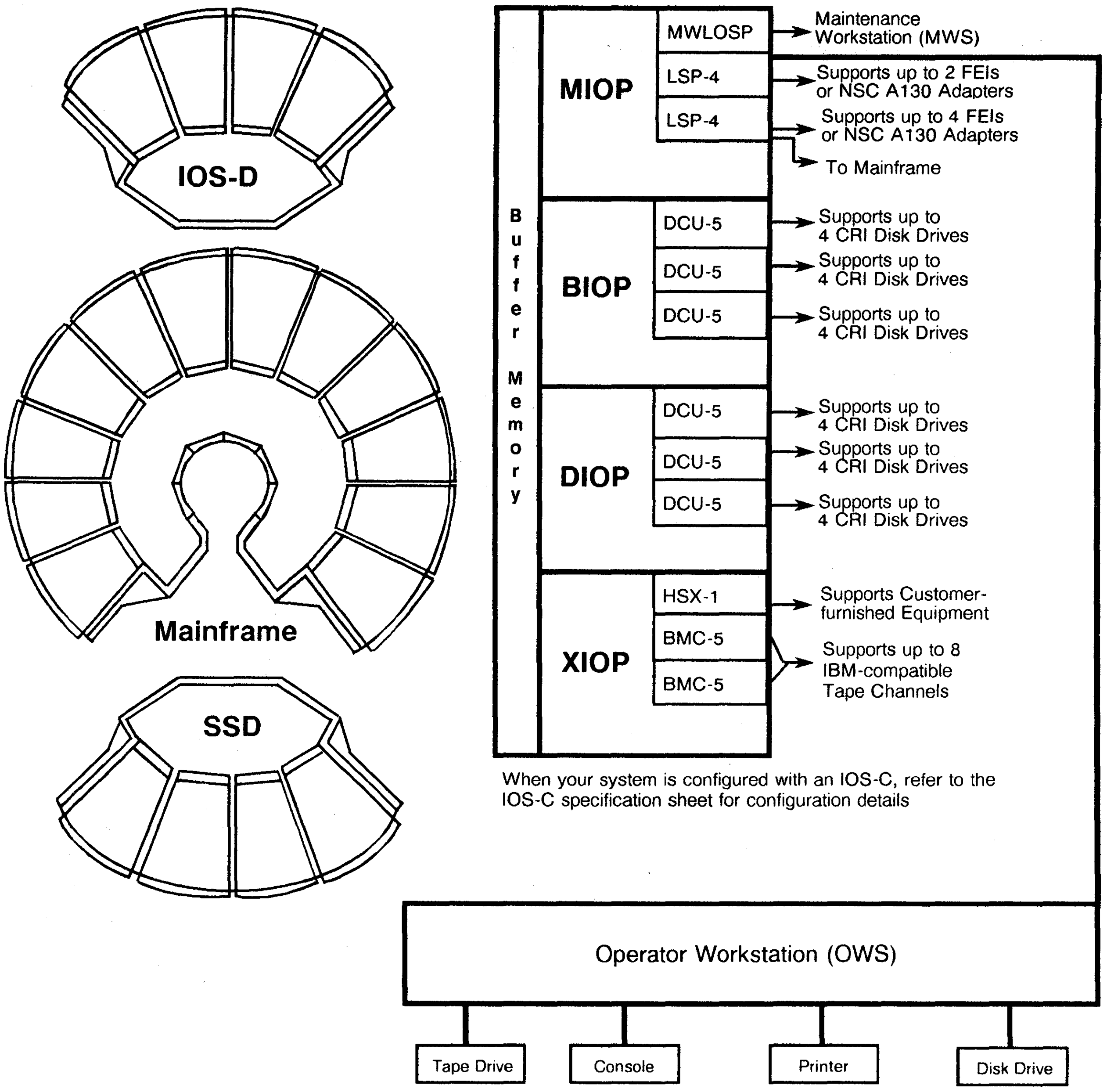
Refrigeration Condensing Unit	1
Motor Generator Sets	2
Operator Workstation	1
Maintenance Workstation	1
Power Distribution Unit	2 or 3

CRAY X-MP EA/4 COMPUTER SYSTEM



Specification Sheet

HR-3020/4


IOS-D
Mainframe
SSD
Buffer Memory
MIOP
MWLOSP
Maintenance Workstation (MWS)
LSP-4
Supports up to 2 FEIs or NSC A130 Adapters
LSP-4
Supports up to 4 FEIs or NSC A130 Adapters
To Mainframe
BIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
Supports Customer-furnished Equipment
BMC-5
BMC-5
Supports up to 8 IBM-compatible Tape Channels
When your system is configured with an IOS-C, refer to the IOS-C specification sheet for configuration details
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

CRAY X-MP EA/4 Computer System Maximum Configuration

CRAY X-MP EA/4 MAINFRAME FEATURES

System Clock

Speed	8.5 ns

CPU Specifications

Number of CPUs	4

Number of Registers per CPU

Address Registers (A) 32 bits each	8
Intermediate Registers (B) 32 bits each	64
Scalar Registers (S) 64 bits each	8
Scalar-save Registers (T) 64 bits each	64
Vector Registers (V) 64 bits per element 64 elements per register	8

Number of Functional Units per CPU

Address Addition	1
Address Multiplication	1
Scalar Addition	1
Scalar Shift	1
Scalar Logical	1
Scalar Population Parity/Leading Zero	1
Vector Addition	1
Vector Multiplication	1
Full Vector Logical	1
2nd Vector Logical	1
Vector Population/Parity	1
Floating-point Addition	1
Floating-point Multiplication	1
Floating-point Reciprocal	1

Shared Resources

Central Memory

Word Width	64 bits
SECDED Error Correction	8 bits
Memory Size	16, 32, 64 Mwords
Number of Banks	32, 64
Number of Ports per CPU	4

I/O Section

1,000 Mbyte/s Channels	2
100 Mbyte/s Channels	4
6 Mbyte/s Channels	4

Shared Registers, Five Clusters Consisting of:

Shared Address (SB) 32 bits each (Y-mode) 24 bits each (X-mode)	8
Shared Scalar (ST) 64 bits each	8
Semaphore (SM) 1 bit each	32

Real-time Clock (64 bits)	1

PHYSICAL DESCRIPTION

Floor Space	42.25 ft^2 (3.5 m^2)
Weight	1,130 lbs (5,126 kgs)
Columns	12

SUPPORT EQUIPMENT

Refrigeration Condensing Unit	2 or 3
Motor Generator Sets	3
Operator Workstation	1
Maintenance Workstation	1
Power Distribution Unit	2 or 3

CONTENTS

3 - I/O SUBSYSTEM

The Cray Research, Inc. (CRI) I/O Subsystem (IOS) provides high-capacity data communications between Central Memory of the CRAY X-MP EA mainframe and peripheral devices, data storage devices, front-end computers, and networks.

The IOS is usually housed in its own stand-alone cabinet. For the CRAY X-MP EA/14se or the CRAY X-MP EA/116se, however, the IOS is housed in the mainframe chassis. Refer to the specification sheet at the end of this section for more specific IOS information.

Each IOS includes multiple I/O Processors (IOPs), a shared memory section (Buffer Memory), channel interfaces, and a network of peripheral equipment dedicated to maintenance operation (refer to Figure 1-2). A single crystal-controlled clock controls the IOS .

The IOS uses an error multiplexer for detecting and reporting IOS system errors. This multiplexer passes channel error information and memory error information to a maintenance computer where the maintenance computer program logs the error information for later analysis.

I/O PROCESSORS

The IOS can contain up to four IOPs. Each IOP is a fast, multipurpose computer capable of transferring data at extremely high rates. A 16-bit processor and bipolar Local Memory combine to support high-speed I/O operations. (Local Memory is independent for each IOP and is different from Buffer Memory, which is shared by all IOPs.) These input and output capabilities make the IOS useful for network control, mass storage access, and computer interfacing.

Each IOP has a control area, a computing area, an I/O area, and a memory area called Local Memory. The following paragraphs give a brief description of each area of the IOS.

The IOP control has an Instruction stack, a Program Exit stack, and control logic; it controls the movement of instructions from memory and decodes them into the appropriate function signals. Instruction codes are run as 1-parcel or 2-parcel instructions; branching and I/O instructions are also included.

Instructions are stored in Local Memory and are transferred into the Instruction stack under the control of the Program Address counter. Instructions issue from the Instruction stack and are then decoded into the appropriate control signals.
The Program Exit stack of the control section stores return addresses for program subroutine calls. The registers provide levels of subroutines in a program. A register keeps track of the levels involved. The IOP computing area contains operand registers, functional units, and an accumulator that work together to run program instructions

stored in memory. The operand registers are used for temporary data storage or indirect memory addressing. The available functional units are a Logical Operation, an Adder, and a Shifter. The accumulator temporarily stores operands or results. All data movement within the IOP uses the accumulator either as a source of data or as the destination for results. The accumulator is also used for all transfers between memory and operand registers.

Functional units in an IOP receive operand pairs and produce single results. One operand address is designated by the instruction, and the other operand is contained in the accumulator. Typically, data flows from Local Memory to the accumulator, from the accumulator (with an operand) to a functional unit, back to the accumulator, and from the accumulator to Local Memory.

An IOP supports channels for input or output use and has direct memory access ports (DMA ports) to Local Memory. Because the channels share the DMA ports, a DMA port may support several channels. The slower the required data rate on the channels, the more channels can be multiplexed into a single DMA port. Each port is bidirectional; input and output channels can be active at the same time as long as they reference different memory area.

Channels use Busy and Done flags to signal the IOP and communicate directly with the IOP accumulator for control information. All channels communicate status and functions through the accumulator. Some low-speed devices transfer data directly to and from the accumulator using one of the channel registers.

Two types of channels are used on an IOP: Accumulator channels and DMA channels. Operating characteristics for the accumulator channels and the channels using DMA ports are similar in many respects.

Accumulator channels can be bidirectional, transferring data to and from the accumulator. They are primarily used to transfer control or status information among the IOPs. Each accumulator channel uses several signals to communicate with the channel interface of other devices.

DMA channels are used for high-speed block transfers and are basically accumulator channels that also allow direct access to an IOP's Local Memory. In addition to the accumulator channel signals, the DMA channel also uses the signals to communicate with the channel interface (channel interfaces are explained later in this section).

The independent memory area for each IOP, is called Local Memory and consists of random access solid-state storage. An error correction and detection network ensures that the data written into memory or read from memory is correct.

I/O Processor Functions

Software in each processor performs specific functions and is structured to perform its tasks efficiently. Each IOP logs information and keeps statistics about channel use and error detection and recovery. The IOPs use Buffer Memory to communicate with each other and to perform functions for one another.

Each IOP is equipped for high-speed I/O transfers between Buffer Memory and Central Memory and communication with the CRAY X-MP EA mainframe.

Each IOP of the IOS runs independently and has its own set of functions. IOP functions are defined by the way the IOP is attached to the other processors (and the mainframe) and by the peripheral equipment attached to the IOP.

One IOP is always designated the Master I/O Processor (MIOP). The MIOP controls the front-end interfaces (FEIs) and the standard group of station peripherals. The MIOP is connected to the Operator Workstation (OWS); this network contains controllers for maintenance peripheral devices. The MIOP is the first IOP to be deadstarted and functions through the accumulator. The MIOP is also connected to Buffer Memory and to the mainframe over a 6-Mbyte/s channel pair.

The IOP designated to interface with the mass storage devices is called the Buffer I/O Processor (BIOP). The BIOP is the main link between the mainframe's Central Memory and the mass storage devices. Data from mass storage devices is transferred back and forth through the BIOP's Local Memory to the mainframe's Central Memory through a 100-Mbyte/s channel pair.

Another IOP that can be added to interface with additional disk storage units (DSUs) is called the Disk I/O Processor (DIOP). The DIOP connects to Buffer Memory and to the mainframe's Central Memory over a 100-Mbyte/s channel pair. The DIOP data transfer sequence is similar to the BIOP's sequence (the DIOP is not available on the CRAY X-MP EA/se mainframes).

The Auxiliary I/O Processor (XIOP) uses block multiplexer controllers (BMCs) to interface between the CRAY X-MP EA computer system and the block mulitplexer channels in an IBM computer system. The XIOP connects to Buffer Memory and to the mainframe's Central Memory over a 100-Mbyte/s channel pair. It also interfaces with the HSX; this channel is used to communicate with customer-furnished peripheral equipment.

I/O Channel Interfaces

To take advantage of an IOP's capabilities, channel interfaces are required to adapt the IOP to other devices. These interfaces buffer data, generate control signals for the device, and multiplex several devices into the same IOP channel.

I/O SUBSYSTEM BUFFER MEMORY

Buffer Memory assists data transfers between peripheral devices and Central Memory of the CRAY X-MP EA mainframe. Refer to the specification sheet at the end of this section for specific Buffer Memory sizes. All IOPs share Buffer Memory, which uses single-error correction/double-error detection (SECDED) data protection. The data is refreshed; refreshing is transparent and does not affect random access capability, although it can cause bank conflicts.

OPERATOR WORKSTATION

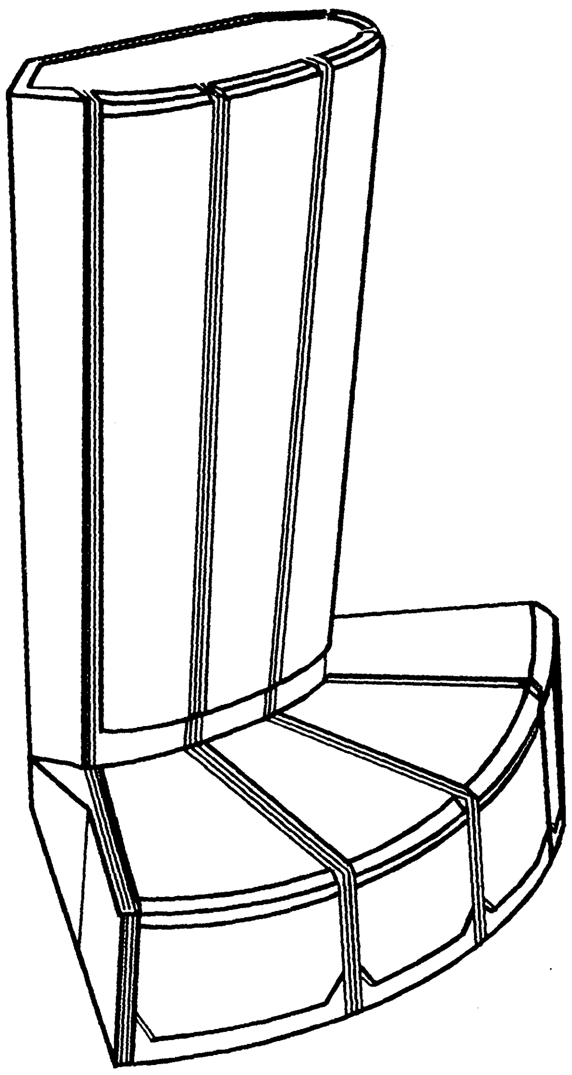
The Operator Workstation (OWS) is a microcomputer system that provides the following functions:

- System operator interface
- System deadstart and master clear functions
- Software maintenance utilities
- Local tape, local removable disk, and local printer
- System time-of-day clock
- Remote access for software and hardware service
- System monitoring functions requiring security

In addition, the VMEbus technology is used in the OWS, the OWS provides an Ethernet interface, which can be used to network workstations in a multiple system site or for multiple system operators.

The devices connected to this OWS include disk drives, a magnetic tape drive, a printer, and an external clock. The OWS communicates with the CRAY X-MP EA computer system through a 6-Mbyte/s channel pair from an IOP located in the IOS. The tape drives, disks, printer, and time-of-day clock are available to the mainframe over this channel.

I/O SUBSYSTEM MODEL C



Specification Sheet

HR-3020/IOS

IOS MODEL C FEATURES

System Clock

Speed	12.5 ns

IOP Specifications

Maximum number of IOPs	4
Number of MIOPs The MIOP supports the front-end interfaces (FEIs) and station software	1
Number of BIOPs The BIOP supports the disk control units (DCU-5) that support the disk drives	1
Number of DIOPs (optional) The DIOP supports the disk control units (DCU-5) that supports the disk drives	1 or 2
Number of XIOPs (optional) The XIOP supports the block multiplexer channel (BMC-5) and HSX-1	1 or 2

Memory

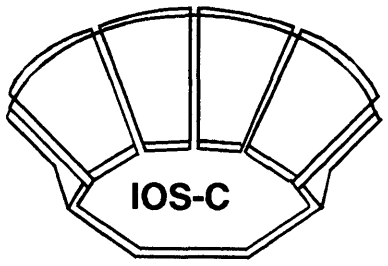
Word Width	64 bits
Buffer Memory size	2, 4, 8, 32 Mwords
Local Memory size per IOP	65 Kwords

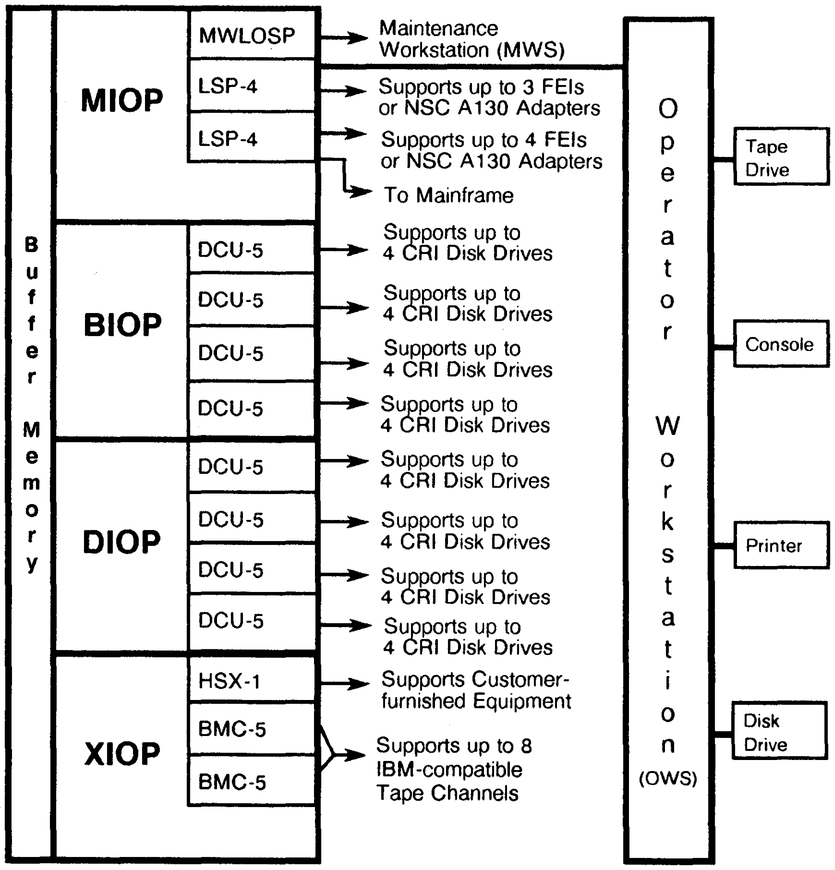
PHYSICAL DESCRIPTION

Floor Space	14.66 ft^2 (1.4 m^2)
Weight	3,220 lbs (1,460 kgs)

SUPPORT EQUIPMENT

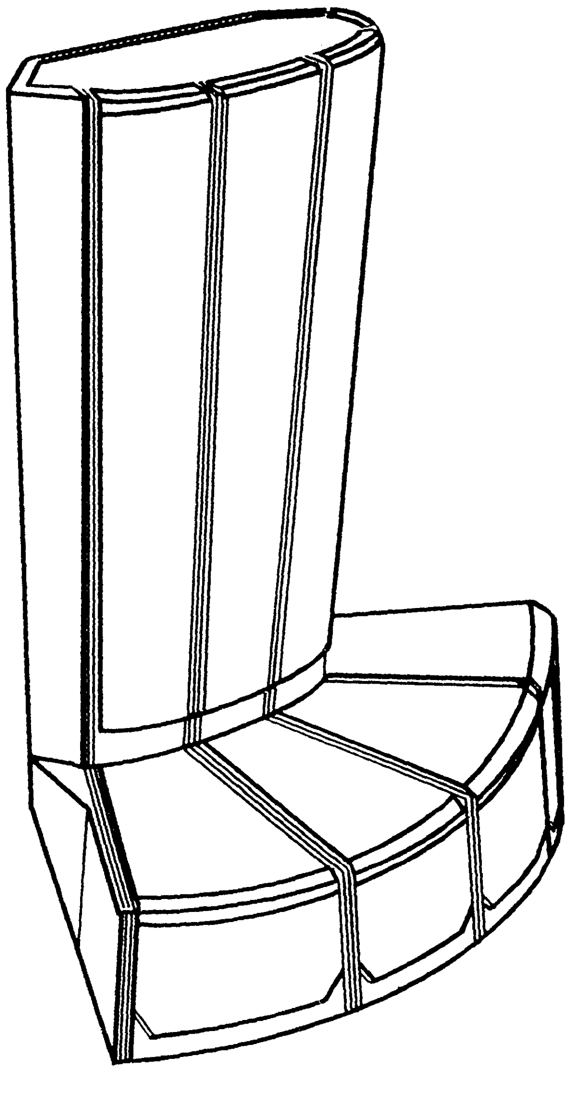
Power Distribution Unit	1
Refrigeration Condensing Unit	1
Maintenance Workstation	1
Motor Generator Set	1




Buffer Memory
MIOP
MWLOSP
Maintenance Workstation (MWS)
LSP-4
Supports up to 3 FEIs or NSC A130 Adapters
LSP-4
Supports up to 4 FEIs or NSC A130 Adapters
To Mainframe
BIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DIOP
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
DCU-5
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
Supports Customer-furnished Equipment
BMC-5
BMC-5
Supports up to 8 IBM-compatible Tape Channels
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

IOS Model C Maximum Configuration

I/O SUBSYSTEM MODEL D



Specification Sheet
HR-3020/IOS

IOS MODEL D FEATURES

System Clock

Speed	12.5 ns

IOP Specifications

Maximum number of IOPs	4
Number of MIOPs The MIOP supports the front-end interfaces (FEIs) and station software	1
Number of BIOPs The BIOP supports the disk control units (DCU-5) that support the disk drives	1
Number of DIOPs (optional) The DIOP supports the disk control units (DCU-5) that supports the disk drives	1 or 2
Number of XIOPs (optional) The XIOP supports the block multiplexer channel (BMC-5) and HSX-1	1 or 2

Memory

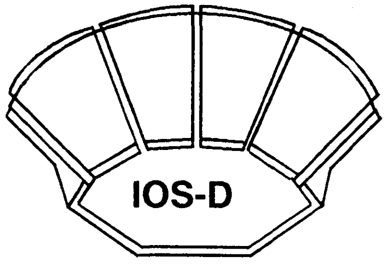
Word Width	64 bits
Buffer Memory size	2, 4, 8, 32 Mwords
Local Memory size per IOP	65 Kwords

PHYSICAL DESCRIPTION

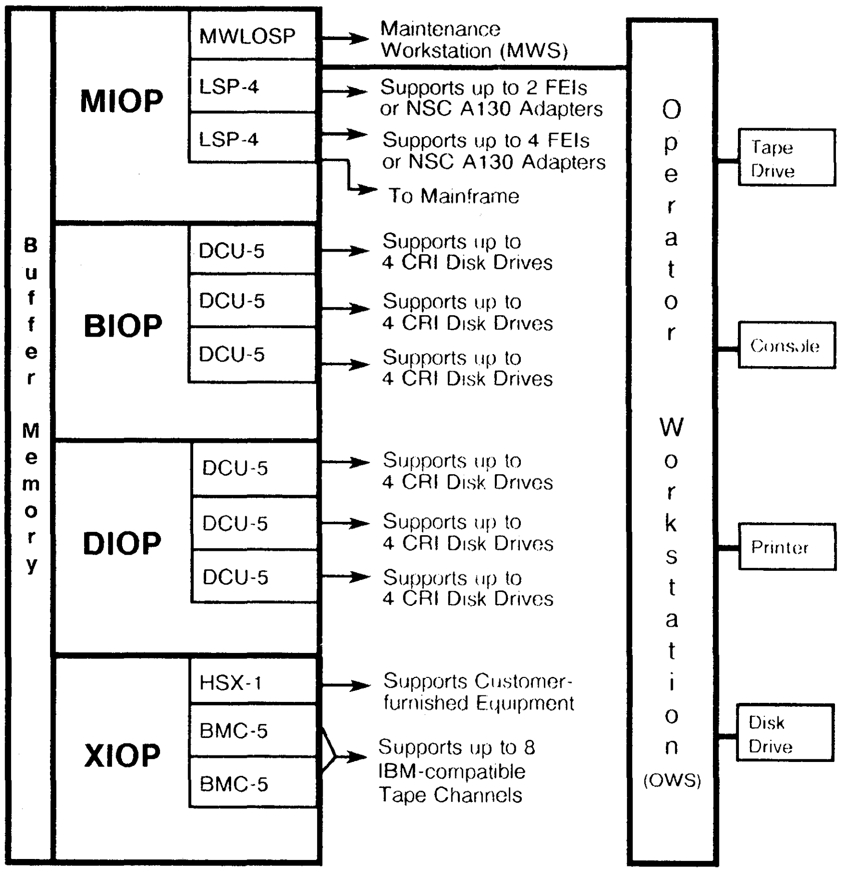
Floor Space	14.66 ft^2 (1.4 m^2)
Weight	3,220 lbs (1,460 kgs)

SUPPORT EQUIPMENT

Power Distribution Unit	1
Refrigeration Condensing Unit	1
Maintenance Workstation	1
Motor Generator Set	1



IOS Model D Maximum Configuration


Buffer Memory
MIOP
MWLOSP
LSP-4
LSP-4
Maintenance Workstation (MWS)
Supports up to 2 FEIs or NSC A130 Adapters
Supports up to 4 FEIs or NSC A130 Adapters
To Mainframe
BIOP
DCU-5
DCU-5
DCU-5
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
DIOP
DCU-5
DCU-5
DCU-5
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
Supports up to 4 CRI Disk Drives
XIOP
HSX-1
BMC-5
BMC-5
Supports Customer-furnished Equipment
Supports up to 8 IBM-compatible Tape Channels
Operator Workstation (OWS)
Tape Drive
Console
Printer
Disk Drive

CONTENTS

4 - SSD SOLID-STATE STORAGE DEVICE

The Cray Research, Inc. (CRI) SSD solid-state storage device is a mass memory storage device similar in usage to a Disk Storage Unit (DSU). Because of its fast access time and large storage capacity, the SSD enhances the performance of Cray computer systems by significantly reducing I/O wait time. The storage media is solid-state, dynamic random access memory (DRAM) chips rather than magnetic film, and the transfer rate is considerably faster than that of a DSU. Datasets are identical to those on disk storage providing portability and flexibility. Maximum transfer rates for the SSD depend on the Cray computer system used and SSD memory size and configuration.

The SSD chassis is shown in Figure 1-3. This self-contained chassis holds the SSD memory modules, channels, and control logic. The power supplies and cooling system are similar to those used in a Cray mainframe. Refer to the specification sheet at the end of this section for the support equipment required for your SSD.

The SSD chassis can be configured with different memory sizes and channel connections, including a channel to the IOS. Refer to the specification sheet at the end of this section for more detailed information.

SSD FUNCTIONS

When an SSD is configured with a Cray mainframe, the SSD connects directly to the mainframe over a special channel or to the IOS. The SSD provides high-speed data transfer to or from Central Memory under the mainframe's software control.

Data is transferred between the SSD buffers and the SSD memory, and between the SSD buffers and the CRAY X-MP EA mainframe. For data transfer, each central processing unit (CPU) memory port controls 16 words through one of two buffers. Each CPU memory port controls every other word to or from memory.

The SSD has five memory ports:

- Port 0 controls refreshing of SSD memory.
- Port 1 connects the SSD to a maintenance computer through a 6-Mbyte/s channel with asynchronous control logic.
- Port 2 provides up to four 100-Mbyte/s channels that may be connected to the IOS. These four channels provide the data connection to peripherals (disk or tape drives and the HSX) which allows data to move to or from the SSD without going through the mainframes Central Memory.
- Ports 3 and 4 connect the SSD directly to the CRAY X-MP EA mainframe using a channel that has a maximum data transfer rate of 1,000-Mbyte/s.

SSD MEMORY SIZE

The SSD has several memory size options; refer to the specification sheet at the end of this section for more information. SSD memory is organized into identical memory groups which are linked logically and used as separate parallel data paths.

SSD MEMORY TRANSFER AND DATA PROTECTION

All transfers into or out of SSD memory are done in 64-word blocks. Individual words are not accessible by addressing. If you provide a starting address for a 64-word block, a full block is read or written. To read a particular word, the entire block is transferred to Central Memory and the word is selected using software methods similar to disk storage units.

SSD addressing is the same for all memory size options; however, different address crossover modules are used to allow for the increase in addressing requirements. SSD memory control logic routes each word to the correct location. Control logic is the same for all memory size options.

To protect data, single-error correction/double-error detection (SECDED) logic is used in SSD memory and on data channels to or from the SSD.

When data is written into SSD memory, a checkbyte (an 8-bit Hamming † code) is generated for the word to be checked and stored with that word. When the word is read from SSD memory, the checkbyte and data word are processed to determine if any bits were altered. If no errors occurred, the word is passed to either the mainframe or the IOS.

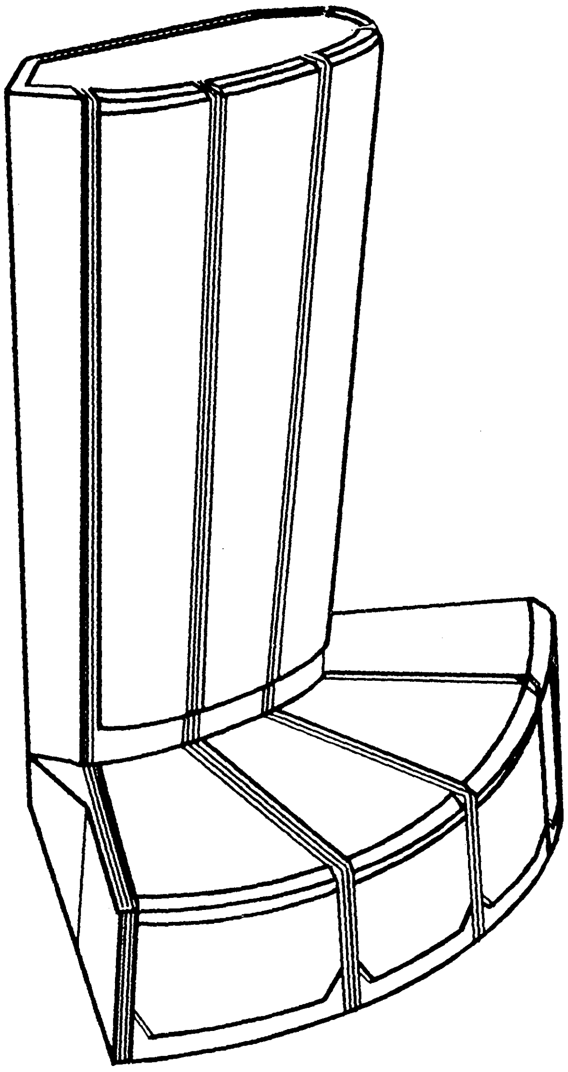
If an error occurred, the 8 bits of the checkbyte are analyzed by the SECDED logic to find the number of altered bits. If only a single bit was altered, the correction logic resets that bit to the correct state and passes the corrected word out to either the mainframe or the IOS. The Error Logger receives details of the error.

If more than a single bit was altered, the SECDED logic cannot correct the word. When a double-bit error is detected, an error flag is set in the A register (Cray computer system configuration), or the Busy and Done flags are both set (IOS configuration), and an error code is generated and sent to the Error Logger. If more than 2 bits are in error, the results are unpredictable.

SECDED is also used on the data channel when writing data into the SSD. Errors that occur on the channel or in Port 2, Port 3, or Port 4 logic are corrected and processed as described. Therefore, there is data protection for write data before storage and data protection for read data after storage.

† Hamming, R. W., "Error Detection and Correcting Codes," *Bell System Technical Journal, 29, No. 2* (April 1950): pages 147-160.

SSD SOLID-STATE STORAGE DEVICE



Specification Sheet

HR-3020/SSD

SSD FEATURES

Storage Capacity	
SSD-3I	32 Mwords[1]
SSD-5I	128 Mwords[1]
SSD-5	128 Mwords
SSD-6	256 Mwords
SSD-7	512 Mwords

Storage Word Size .... 72 bits
(64 data bits and 8 check bits)

Minimum Block Size .... 64 words

Maximum Blocks per Operation	
Port 2	16 Kwords
Ports 3 and 4	16 Mwords

Data Transfer Burst Speeds	
Port 2	100 Mbyte/s
Ports 3 and 4	1,000 Mbyte/s

Available Channels	
Port 1 (6 Mbyte/s)	1
Port 2 (100 Mbyte/s)	4[2]
Port 3 (1,000 Mbyte/s)	1
Port 4 (1,000 Mbyte/s)	1[3]

[1] The SSD-3I and SSD 5I are housed in the IOS-C or D Chassis
[2] The SSD-3I and SSD 5I only supports 2 ports
[3] The SSD-3I and SSD 5I do not support Port 4

Maximum Band Width .... 26.9 Gbits/s

Error Correction
Single-error correction/double-error detection (SECDED) before and after storage and on all user channels

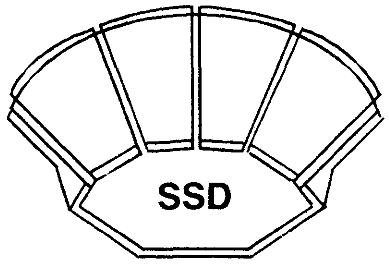
Port Priorities
Lowest port number has highest priority

PHYSICAL DESCRIPTION
(Stand-alone SSD Only)

Floor Space	14.66 ft^2 (1.4 m^2)
Weight	3,220 lbs (1,460 kgs)
Columns	4

SUPPORT EQUIPMENT
(Stand alone SSD Only)

Power Distribution Unit	1
Motor Generator Set	1



SSD Model	Size (Mwords)	Size (Mbytes)	Size(64-bit words)
SSD-3I	64	512	67,108,864
SSD-5I	128	1,024	134,217,728
SSD-5	128	1,024	134,217,728
SSD-6	256	2,048	268,435,456
SSD-7	512	4,096	536,870,912

SSD Models and Sizes

CONTENTS

5 - PERIPHERAL EQUIPMENT

The following subsections describe the major components of the disk drives and various network interfaces used with the CRAY X-MP EA computer system.

DISK CONTROLLER UNITS AND DISK STORAGE UNITS

A disk system provides long-term intermediate data storage for a Cray computer system. Components of the disk system include: Buffer I/O Processor (BIOP) and/or the Disk I/O Processor (DIOP) of the I/O Subsystem (IOS), Disk Controller Units (DCUs), and Disk Storage Units (DSUs). Each BIOP and DIOP can be configured with a minimum or maximum number of DCUs and DSUs; refer to the appropriate disk drive specification sheet at the end of this section for the configuration information. Refer to Section 3 of this manual for a description of the IOS and its IOPs.

The DCU modules are contained in the IOS chassis and are connected to IOP channels. Each DCU requires one DMA port and one to four accumulator channels from the IOP.

Disk Controller Units

DCUs are housed in the IOS and are the interface between the IOP and the disk drives. The DCUs consist of logic modules for data transfer, buffer storage, and control. An interface between the DCU and the DSU transfers parcel size parameters, statuses, and data. Head deskew, data assembly, and data disassembly are controlled in the disk drive interface logic. Data and some statuses are transferred in 16-parcel packets.

The DSUs are controlled by channel functions from an IOP to a DCU. The DCU interprets the functions and generates the proper control signals for the DSU. Status is returned by the DSUs to registers in the DCU where it can be returned to the IOP's accumulator through the proper channel function.

Disk Storage Units

The DSUs store data on magnetic disks. Flaw handling mechanisms are provided so that the operating system has fewer flaws to track.

A DSU consists of several rotating platters. Data is accessed by read/write heads organized into groups. Heads are controlled and positioned by one or more head actuator (servo) mechanisms to the disk cylinders.

The recording surface available to each head group is called a disk track, which is the basic storage unit reserved by the operating system. Each disk track has sectors where data is recorded and read back. The data in one sector is called a data block and includes verification and error-correction data. Data can be transferred between an IOP's Local Memory and the disk surface only in these data blocks (512 64-bit words).

The following subsections describe the specific DSUs.

DD-39 Disk Storage Unit

The DD-39 DSU, hereafter referred to as the DD-39, includes three disk drives and the required interface logic to operate as a single disk drive unit. Each disk drive within the cabinet consists of six rotating platters. Data is accessed by 20 read/write heads organized into five groups. Heads are controlled and positioned by a servo mechanism to one of 842 disk cylinders.

The recording surface available to each head group is called a disk track, which is the basic storage unit reserved by the operating system. Each disk track has 24 sectors (and one spare sector) where data is recorded and read back. The data in one sector is called a data block and consists of 2,048 16-bit parcels of IOP data (512 64-bit words) plus verification and error-correction data. Data can be transferred between the IOP's Local Memory and the disk surface only in data blocks of 2,048 16-bit parcels. Sectors may be chained for both read and write operations. The DD-39 responds to commands from the IOP.

Interface logic in the DD-39 cabinet that adapts the DCU-5 signals and protocol to the individual drive units, coordinates routing among the drives, and buffers the drive data. The interface logic performs the following:

- Controls unit selection among the three drives
- Passes control functions to the selected drive
- Buffers read and write data for transfers
- Generates error correction codes for write data
- Checks read data correction codes and corrects read data if necessary

The DD-39 has a sector-slipping mechanism that allows a full track to remain available to the system even after one sector of the track becomes flawed. Sectors are slipped from the flawed sector to the end of the track. In general, if sector n becomes flawed, sectors n through 23 of the track are slipped, and the data contained in these sectors must be recreated. If a second sector in a track becomes flawed, the operating system must mark the sector as unavailable. Sector slipping takes place off-line. A hardware diagnostic routine reformats the track with slipped sectors.

A DD-39 has 25 sectors per track, although only 24 sectors are used for data. Under normal circumstances the spare sector is ignored. If one of the data sectors becomes flawed however, the spare sector is used as a data sector.

Refer to the DD-39 Disk Storage Unit Specification Sheet at the end of this section for configuration information.

DD-49 Disk Storage Unit

The DD-49 DSU, hereafter referred to as the DD-49, includes nine rotating platters. Data is accessed by 32 read/write heads organized into eight groups with four read/write heads per group. Heads are controlled and positioned by two identical head actuator (servo) mechanisms to one of 886 disk cylinders. The servo mechanisms are identified as Servo-A and Servo-B.

The recording surface available to each head group is called a disk track, which is the basic storage unit reserved by the operating system. Each disk track has 42 sectors (and two spare sectors) where data is recorded and read back. The data in one sector is called a data block and consists of 2,048 16-bit parcels of IOP data (512 64-bit words) plus verification and error-correction data. Data can be transferred between the IOP's Local Memory and the disk surface only in blocks of this size. Sectors may be chained for both read and write operations.

The DD-49 responds to commands from the IOP through a microprocessor unit card (MPU card) that contains a 68000 type 16-bit microprocessor and a second processor called the Supervisor. The DD-49 requires a DCU-5.

The DD-49 provides a sector-slipping mechanism that allows a full track to remain available to the system even after one or two sectors of the track become flawed. Sectors are slipped from the flawed sector to the end of the track. In general, if sector n becomes flawed, sectors n through 41 of the track are slipped, and the data contained in these sectors must be recreated. If a second sector in a track becomes flawed, the process is repeated. If a third sector in a track becomes flawed, the operating system must mark the sector as unavailable. Sector slipping takes place off-line. A hardware diagnostic reformats the track with slipped sectors.

A DD-49 has 44 sectors per track, 42 of which are used for data. Under normal circumstances the two spare sectors are ignored; if one of the data sectors becomes flawed, however, a spare is used as a data sector.

Refer to the DD-49 Disk Storage Unit Specification Sheet at the end of this section for configuration information.

DS-40 Disk Subsystem

The DS-40 Disk Subsystem includes the following components: the DD-40 Disk Storage Unit, hereafter referred to as the DD-40, the DC-40 Disk Control Unit hereafter referred to as the DC-40, and the Disk Controller Cabinet (DCC-2). The DD-40 contains four disk drives and the required interface logic to operate as a single disk drive unit. The DC-40 is housed in the DCC-2, which is separate from the DD-40 disk drives. Refer to the DS-40 Disk Subsystem Specification Sheet at the end of this section for the exact configuration information. Each disk drive consists of six rotating platters and ten recording surfaces. Data is accessed by 19 read/write heads controlled and positioned by a servo mechanism to one of 1,418 disk cylinders.

The recording surface available to each head is called a disk track, which is the basic storage unit reserved by the operating system. Each disk track has 48 sectors where data can be recorded and read back. The data in one sector is called a data block and consists of 2,048 16-bit parcels of IOP data (512 64-bit words) plus verification and error-correction data. Data can be transferred between the IOP's Local Memory and the disk surface only in blocks of this size. Sectors may be chained for both read and write operations.

Interface logic in the DC-40 adapts to the DCU-5 signals and protocol to the individual disk drive units, handles routing among the drives, and buffers the data from the four drives in a full-track buffer. The interface logic in a set of four DC-40s performs the following:

- Controls up to 32 drives
- Passes control functions to the selected drives
- Passes status from the drives to the DCU-5
- Buffers read and write data for transfers between DCU-5 and the disk drives
- Generates error-correction codes for write data
- Checks read data correction codes and corrects read data if necessary (these corrections are completed in the IOP)
- Controls distribution of read/write data over 48 sectors per track using 12 sectors from each of the four drives

The DC-40 provides a method for detecting flaws in the recording media and then avoids these flawed areas during a read or write operation. A factory flaw table is used initially; if any additional flaws are found, diagnostic programs determine where the flaw is located in the sector and how wide it is.

When writing data into a flawed sector, the DC-40 writes data onto the sector until the defective location is reached. In the area starting at the defect address, the DC-40 writes a 16-byte field of 0s. Following this field of 0s, the DC-40 resumes writing data. When reading data into a flawed sector, the DC-40 reads the defect address to locate the beginning of the field of 16 bytes of 0s starts. This field is then skipped and not allowed into the read data. Normal reading begins again after the defect field is passed.

COMMUNICATION INTERFACES

The CRAY X-MP EA computer system can be connected to a wide variety of computer systems (often referred to as "front-end systems") and networks through the IOS. This enables users of non-Cray computer systems to make full use of the CRAY X-MP EA system's extraordinary computing power. The following subsections describe the methods used to interface the CRAY X-MP EA computer system with other computer systems and networks.

FEI-1 Front-end Interface

The FEI-1 front-end interface provides communication between the Cray mainframe and many different types of front-end computer systems. The FEI-1 compensates for differences in channel widths, machine word size, electrical logic levels, and control protocols. Refer to the Front-end Interface Specification Sheet at the end of this section for a complete list of compatible mainframes and minicomputers. The FEI-1 is housed in a stand-alone cabinet located near the host computer. The cabinet is air-cooled and operates directly from the AC power mains; power consumption varies with each type of interface. Internal power supplies provide all required voltages. Cabinet grounding is flexible and can be configured to specific site requirements.

Each FEI-1 contains two or more logic modules and the appropriate cabling. The FEI-1 connects to a channel pair on the LSP-4 in the MIOP. The hardware logic contained in these modules performs all command translation and protocol conversion needed to transfer data; these operations are invisible to both the front-end and Cray programmer.

Fiber-optic Link

The Cray Research, Inc. (CRI) fiber-optic link (FOL) is used as a channel extender for 6-Mbyte/s (16-bit asynchronous) channels. It replaces the conventional wire cabling between a Cray computer system and an FEI-1 with fiber-optic cables. Fiber-optic cabling enhances the performance of the FEI-1 by eliminating the occasional problems related to system isolation, including induced noise, variable ground potentials, and radio frequency radiation found in wire cabling. Fiber-optic cabling overcomes these problems and, in addition, provides a secure link for transmitting data over distances up to 3,280 ft (1,000 m).

Fiber-optic technology uses thin glass fibers (optical fibers) to transmit information from one location to another. Optical fibers are used in place of wire cabling, and light signals replace electrical charges sent over conventional wire cabling.

The FOL operates by converting digital data into electrical pulses. The electrical signal is used to modulate light coming from a light-emitting diode (LED). The resulting light pulses, which are of the same duration as electrical pulses, are sent over the fiber-optic cable. At the receiving end, the light pulses are converted back into electrical pulses, which are then demodulated to recover the digital data. As with a standard FEI-1, these operations are invisible to both the front-end and Cray programmer.

The fiber-optic FEI-1 cabinet is similar to the standard FEI-1 cabinet; it is modified with an attached compartment to hold the fiber-optic modules. In addition to this FEI-1 cabinet, another cabinet containing a complementary set of fiber-optic modules is located next to the Cray mainframe. These special fiber-optic modules modulate and demodulate the signals between the Cray mainframe and the front-end system.

FEI-3 Front-end Interface

The FEI-3 is a group of front-end interfaces that enables certain VME-based microcomputers and workstations to communicate with a Cray system over a standard 6-Mbyte/s I/O channel. Specific FEI-3 applications depend on the capabilities of the VME workstations or microcomputers. Other FEI-3 applications are the following:

- Connection to a communications gateway for Ethernet or other networks
- Connection to a graphics output processor or device
- Connection to a remote Cray station

Each FEI-3 interface consists of two VME-compatible circuit boards that install into the target VME system, plus supporting cables and software drivers. The FEI-3 connects to a channel pair on the LSP-4 in the MIOP. The customer furnishes and provides support for the target VME system.

The VMEbus is an industry standard which specifies the electrical and mechanical rules for a microcomputer backplane. Connections to specific vendor models of the VME-based computer systems are supported by the FEI-3.

Direct Network Connections

The IOS supports direct connection to network adapters such as Network Systems Corporation (NSC) HYPERchannel adapters, Computer Network Technology LANlord adapters, and others. Direct connection to such network adapters occurs via the Master I/O Processor (MIOP) of the IOS (using a channel pair on the LSP-4).

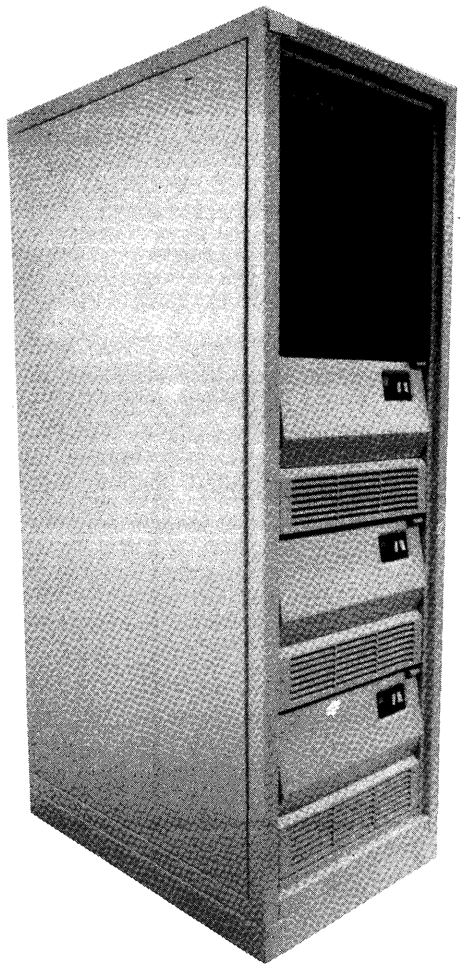
High-speed External Communications Channel

The Cray High-speed External (HSX) Communications channel provides full duplex point-to-point communications between a CRAY X-MP EA computer system and a user-supplied device. The channel operates at up to 100 Mbytes/s and could be used to network multiple Cray systems or to communicate with a very fast graphics output device. The HSX channel is configured through the XIOP.

DEC VAX Supercomputer Gateway

Digital Equipment Corporation (DEC) offers a VAX Supercomputer Gateway to allow direct connection between the DEC VAX cluster environment and a CRAY X-MP EA computer system. This connection is through the LSP-4 channel pair on the MIOP.

DD-39 DISK DRIVE



Specification Sheet
HR-3020/DD-39

DD-39 FEATURES

Storage Capacity	1,284 Mbytes (formatted)
Transfer Rate	
Burst rate	59.5 Mbytes/s
Sustained rate	7.4 Mbytes/s
Total Data Sectors	100,800
Total Data Words	51,609,600
Typical Position Delays	
Single track	5.5 ms
Average	18 ms
Full stroke	35 ms

POWER & COOLING SPECIFICATIONS

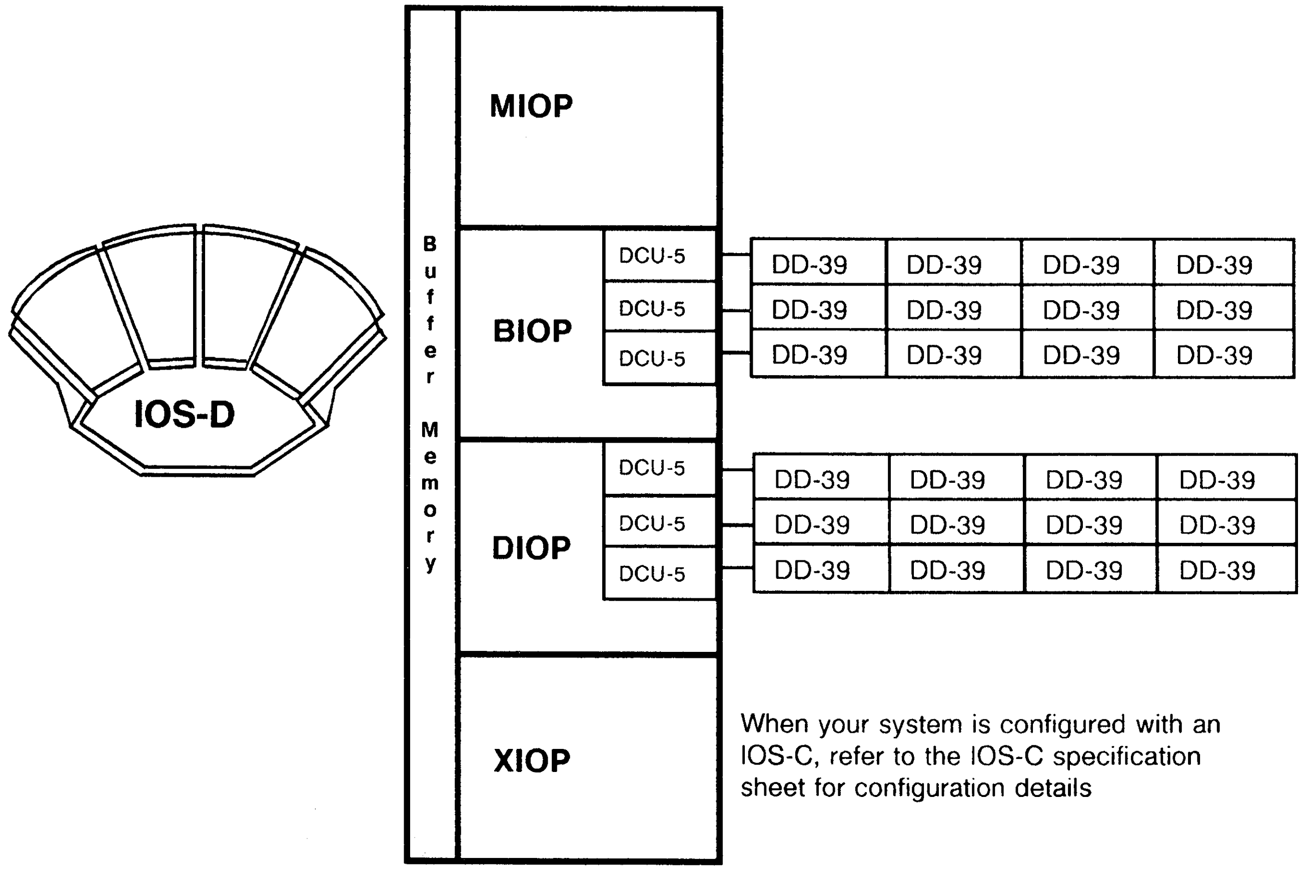
Required Power	3-phase 208 Vac 60 Hz, 20 A
Heat Load	9,520 BTU/hr 2,800 W
Type of Cooling	Air Cooled

PHYSICAL DESCRIPTION

Floor Space	6.0 ft^2 (0.56 m^2)
Weight	1,150 lbs (522 kgs)

PLACEMENT/CABLING SPECIFICATIONS

Minimum Clearance	
Sides	1 in. (2.54 cm)
Front	36 in. (91 cm)
Back	36 in. (91 cm)
Length of Power Cable	10 ft (3 m)
Maximum Length of Data Cables	50 ft (15 m)



When your system is configured with an IOS-C, refer to the IOS-C specification sheet for configuration details

IOS Model D and DD-39 Disk Drive Maximum Configuration

DD-49 DISK DRIVE

Specification Sheet
HR-3020/DD-49

DD-49 FEATURES

Storage Capacity	1,219 Mbytes (formatted)
Transfer Rate	
Burst rate	12 Mbytes/s
Sustained rate	9.6 Mbytes/s
Total Data Sectors	297,696
Total Data Words	152,420,352
Typical Position Delays	
Single track	2.5 ms
Average	16 ms
Full stroke	30 ms

POWER & COOLING SPECIFICATIONS

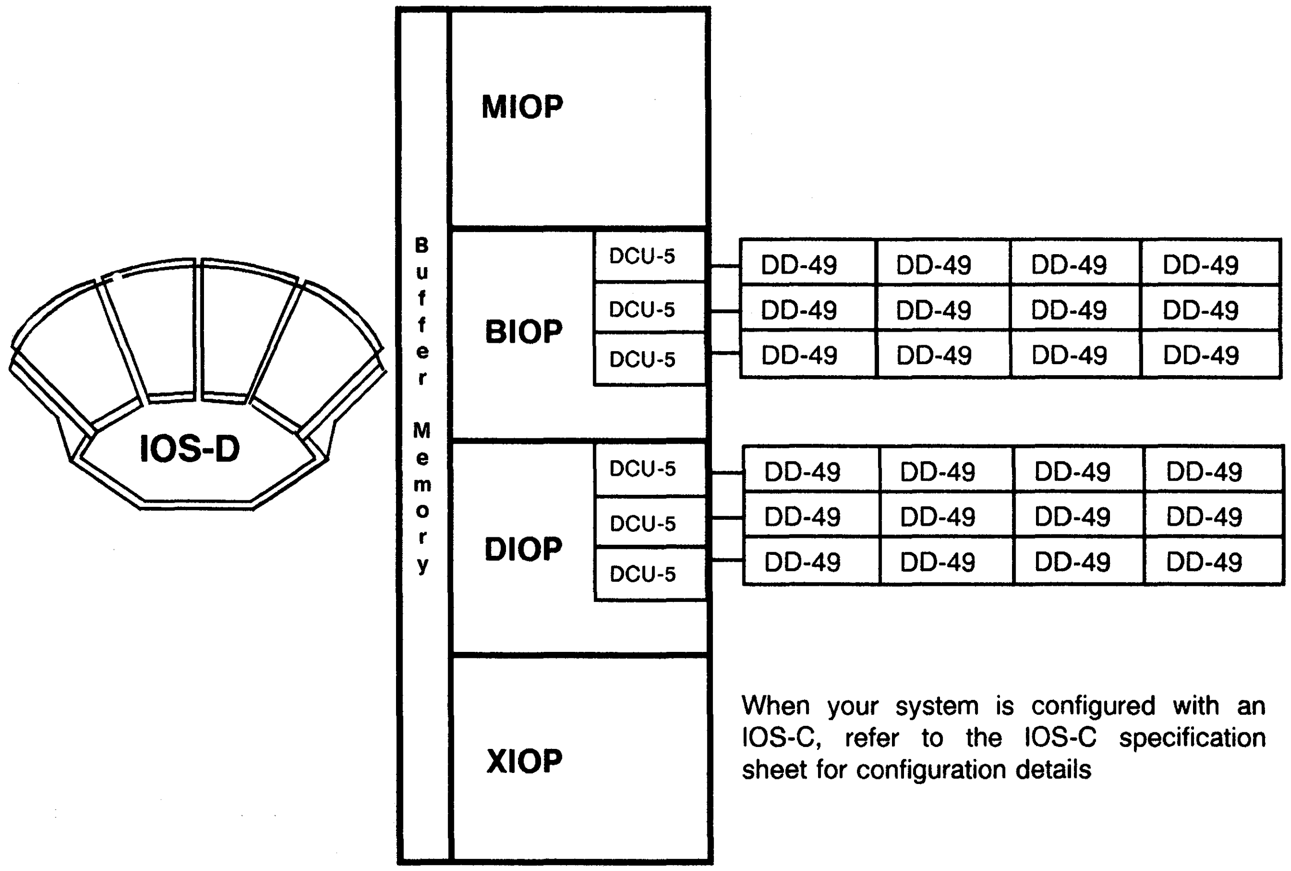
Required Power	3-phase 208 Vac 60 Hz, 20 A
Heat Load	8,200 BTU/hr 2,400 W
Type of Cooling	Air Cooled

PHYSICAL DESCRIPTION

Floor Space	7.4 ft^2 (0.68 m^2)
Weight	900 lbs (408 kgs)

PLACEMENT/CABLING SPECIFICATIONS

Minimum Clearance	
Sides	1 in. (2.54 cm)
Front	36 in. (91 cm)
Back	36 in. (91 cm)
Length of Power Cable	10 ft (3 m)
Maximum Length of Data Cables	50 ft (15 m)



When your system is configured with an IOS-C, refer to the IOS-C specification sheet for configuration details

IOS Model D and DD-49 Disk Drive Maximum Configuration

DS-40 DISK SUBSYSTEM

Specification Sheet
HR-3020/DS-40

DC-40 FEATURES

Transfer Rate

Burst rate	20 Mbytes/s
Sustained rate	9.6 Mbytes/s

POWER & COOLING SPECIFICATIONS

Required Power	3-phase 208 Vac 60 Hz, 60 A
Type of Cooling	Water cooled refrigeration/Air cooling
Water Temperature	40 °F (4.4 °C) to 90 °F (32.2 °C)
Heat Load (to air)	3,054 BTU/hr 895 W
Heat Rejection to Water	24,000 BTU/hr 7,643 W
Storage Capacity	5,200 Mbytes

DD-40 FEATURES

Transfer Rate

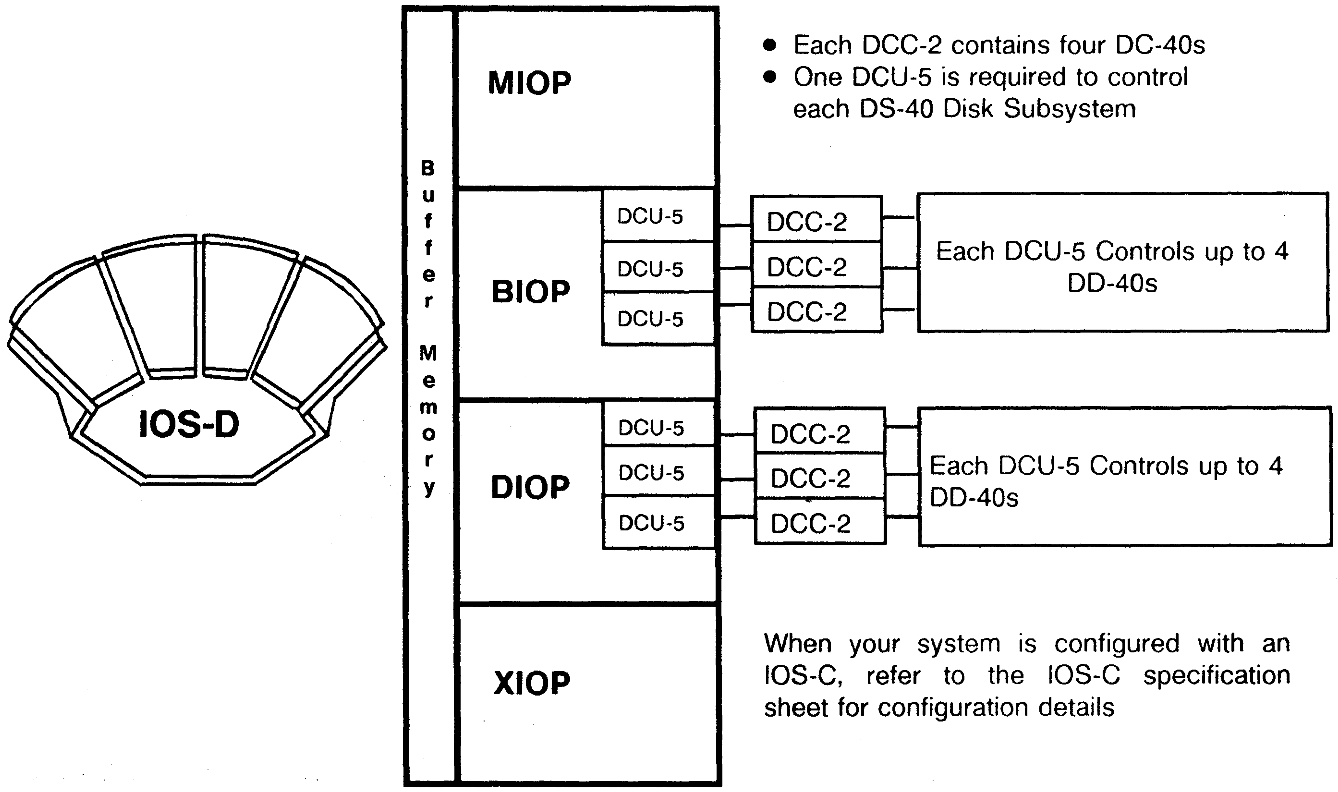
Burst rate	20 Mbytes/s
Sustained rate	9.6 Mbytes/s
Total Data Sectors	1,293,216
Total Data Words	662,126,592

Typical Position Delays

Single track	4 ms
Average	16 ms
Full stroke	30 ms

POWER & COOLING SPECIFICATIONS

Required Power	3-phase 208 Vac 60 Hz, 20 A
Type of Cooling	Air Cooled
Heat Load	8,000 BTU/hr 2,340 W



- Each DCC-2 contains four DC-40s
- One DCU-5 is required to control each DS-40 Disk Subsystem

When your system is configured with an IOS-C, refer to the IOS-C specification sheet for configuration details

IOS Model D and DS-40 Disk Subsystem Maximum Configuration

DCC-2/DC-40 PHYSICAL DESCRIPTION

The DC-40s are housed in a Disk Control Cabinet (DCC-2) that contains the power control and refrigeration components required for the DC-40.

Floor Space	8.6 ft^2 (0.81 m^2)
Weight	1,100 lbs (499 kgs)

DCC-2/DC-40 PLACEMENT/CABLING SPECIFICATIONS

Minimum Clearance	
Sides	12 in. (30 cm)
Front	36 in. (91 cm)
Back	36 in. (30 cm)
Length of Power Cable	8 ft (2.4 m)
Maximum Length of Data Cables	40 ft (12 m)

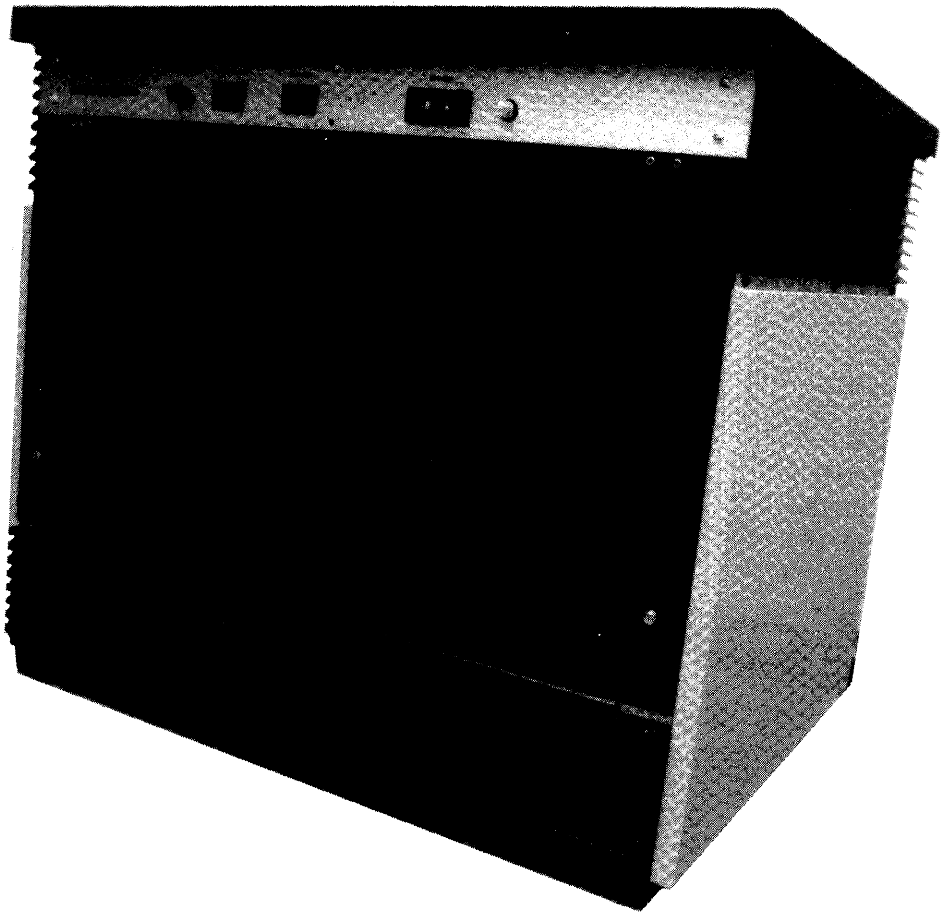
DD-40 PHYSICAL DESCRIPTION

Floor Space	7.2 ft^2 (0.68 m^2)
Weight	1,150 lbs (522 Kgs)

DD-40 PLACEMENT/CABLING SPECIFICATIONS

Minimum Clearance	
Sides	1 in. (2.5 cm)
Front	36 in. (91 cm)
Back	30 in. (91 cm)
Length of Power Cable	6 ft (1.8 m)
Maximum Length of Data Cables	15 ft (4.5 m)

FRONT-END INTERFACE



Specification Sheet
HR-3020/FEI

FEI FEATURES

Cray Research, Inc. (CRI) offers hardware interfaces and station software to connect the CRAY X-MP EA systems to a wide variety of popular computer systems, networks, and workstations.

Mainframes

Amdahl 470 series
CDC 70
CDC 170
CDC 180
CDC 6000
CDC 7600
Honeywell 6000
IBM 360
IBM 370
IBM 303x
IBM 308x
IBM 43xx
Siemens
UNISYS 1100/80 series

Minicomputers and Microcomputers

Data General ECLIPSE series
DEC PDP/11
DEC VAX 11/750
DEC VAX 11/780
DEC VAX 11/782
DEC VAX 11/785
DEC VAX 8600
DEC VAX cluster
Motorola DELTA series microcomputer

Networks

Ethernet (TCP/IP) networks
(with the use of a VME front end interface as a gateway)

Workstations

Sun-3 (through FEI-3s interface)

Operating Systems

Apollo AEGIS
CDC NOS, NOS/BE, and NOS/VE
Data General AOS
Data General RDOS
DEC VAX/VMS
Honeywell Bull
IBM MVS and VM
UNISYS
UNIX

PHYSICAL DESCRIPTION

Floor Space	3.6 ft^2 (0.335 m^2)
Weight	200 lbs (91 kgs)
Height	23 in. (58.4 cm)

CONTENTS

6 - SOFTWARE OVERVIEW

The CRAY X-MP EA computer system comes with a variety of software including the Cray operating systems UNICOS and COS. Two Fortran compilers, CFT77 and CFT, provide automatic vectorizing, as do the C and Pascal compilers. Extensive library routines, multitasking facilities, program- and file-management utilities, debugging aids, and a powerful Cray assembler are included in the system software. A large number of third-party and public-domain application programs also run on Cray systems.

The CRAY X-MP EA computer system is supported by the communications software described later in this section. The TCP/IP protocol, a widely accepted protocol for interconnecting UNIX systems, and Cray proprietary station products used for connecting other vendors' systems and workstations to Cray computers are included in the communications software available.

A list of software publications is provided at the end of this section.

OPERATING SYSTEMS

The CRAY X-MP EA computer system comes with the state-of-the-art operating system UNICOS. The Cray operating system COS is also available on the CRAY X-MP EA computer system. UNICOS and COS provide compatibility with software written for the earlier models of the CRAY X-MP computer system. The operating systems are discussed in the following subsections.

UNICOS

The Cray operating system UNICOS provides powerful interactive and batch capabilities and multiple methods to accomplish a task. These characteristics enable UNICOS to provide exceptional problem solving capabilities that complement the computing capacity of the underlying CRAY X-MP EA computer system. UNICOS efficiently manages high-speed data transfers between the CRAY X-MP EA system and peripheral equipment. UNICOS is written in the high-level language C and is easily ported from one Cray system to another.

UNICOS contains a kernel plus a large set of utilities and library programs. The kernel is the central portion of the system and is a simple, well-constructed structure with short and efficient software control paths. The kernel supports many system call primitives that library and application programs can use together to perform more complex tasks.

Hardware architecture has its primary impact on the UNICOS kernel; the relatively small kernel allows Cray Research, Inc. (CRI) to quickly provide the software that enables the user to make full use of the maximum performance benefits offered by new hardware architectural features.

UNICOS offers a large set of utility programs that allow the user to interact with the operating system. For example the Source Code Control System tracks modifications to files. This system is useful when programs and documentation undergo frequent changes because of development, maintenance, or enhancement. Line-oriented and screen-oriented text editors offer versatility for users wishing to create and maintain text files. In addition to the utilities, UNICOS provides a number of products specifically designed for Cray computer systems. UNICOS supports optimizing, vectorizing Fortran compilers; an optimizing, vectorizing Pascal compiler; and an optimizing, vectorizing C compiler.

UNICOS and UNIX are essentially the same in philosophy, structure, and function; however, CRI has enhanced UNICOS to make full use of the power of the Cray computer system. Enhancements include I/O capabilities to improve supercomputer performance, added multiprocessor and multitasking support, batch processing through the Network Queueing System (NQS), job recovery, additional networking software, accounting features, and others.

COS

The Cray operating system COS operates on the CRAY X-MP EA computer system. COS is a multiple processor, multiprogramming, and multitasking operating system. It offers a batch environment to the user and supports interactive jobs and data transfers where these are available through the front-end system.

COS is written in a modular form, providing the ability to easily tailor its capabilities to meet installation-dependent requirements for security and accounting. The COS data management capability allows for highly efficient creation and maintenance of temporary and permanent datasets.

For users making a transition from COS to UNICOS, the Guest Operating System (GOS) feature of COS allows users to concurrently run COS and UNICOS on the CRAY X-MP EA system. A maximum of 4 CPUs and 16 Mwords of Central Memory can be dedicated to COS. A variety of additional migration tools have been created to further ease conversion from COS to UNICOS.

ON-LINE DIAGNOSTIC SYSTEM

The on-line diagnostic system provides diagnostics that run under the Cray operating systems COS and UNICOS. The on-line diagnostic system performs error detection and ioslation concurrent with the system operation. This type of on-line maintenance provides the following benefits:

- Offers customer an enhanced level of continuous system operation
- Prevents system software failures and identifies data integrity problems in system output

- Reduces the mean time to repair (MTTR) by isolating the failing hardware while the system is running

- Reduces off-line preventive maintenance (PM) time required for failure detection, ioslation, and repair

To ensure maximum system reliability, the on-line diagnostic programs detect, isolate, and report hardware faults gather and analyze system performance data. The following types of on-line diagnostic programs are included in the on-line diagnostic system:

- Confidence tests: These tests provide error detection and isolation

- Maintenance tests: These tests provide error isolation at the functional unit, module, or peripheral level

- Network test: This test exercises all or part of the path between a Cray mainframe and support front-end computer system

- Down-device programs: The down-device programs provide on-line CPU and peripheral testing while the hardware is removed from system operation

- I/O Subsystem (IOS) deadstart programs: These programs can run prior to system deadstart to verify the integrity of the IOS hardware

- Utility programs: These programs are system tools

MULTITASKING

Multitasking is a technique whereby an application program can be partitioned into independent tasks that can run in parallel on different CPUs of the CRAY X-MP EA computer system. This results in a substantial throughput improvement over serially executed programs. To achieve this, two methods can be used: macrotasking and microtasking.

Macrotasking

Macrotasking is an implementation of multitasking that allows parallel execution of code at the subroutine level on multiple-processor systems. Macrotasking is best suited to programs with large long-running tasks. The user interface to the CRAY X-MP EA system's macrotasking capability is a set of Fortran-callable subroutines that clearly define and synchronize tasks at the subroutine level. These subroutines are compatible with similar subroutines available on other Cray products.

Microtasking

Microtasking is an implementation of multitasking that allows parallel execution of very small segments of code on multiple processors. An example of this would be individual iterations of DO loops. With microtasking, the programmer can include compiler directives to further enhance performance beyond the automatic vectorization done by the compiler.

In addition to working efficiently on parts of programs where the granularity is small, microtasking works well when the number of processors available for the job is unknown or may vary during the program's execution. In a batch environment where processors are available for short periods, the microtasked job can dynamically adjust to the number of available processors.

FORTRAN COMPILERS

CRI offers two Fortran compilers for the CRAY X-MP EA computer system: CFT77 and CFT. Both compilers comply with the ANSI 78 (Fortran 77) standards and offer a high degree of automatic scalar and vector optimization. Both accept many nonstandard constructs written for other vendors' compilers. Vectorized object code is produced from standard Fortran code; users can program standard syntax to access the full power of the CRAY X-MP EA system architecture.

CFT77

The Fortran compiler CFT77 is a multipass optimizing portable compiler that processes existing standard Fortran programs, often without modification. CFT77 uses two basic techniques to improve the execution time of a Fortran program: vectorization and scalar optimization.

The compiler automatically generates code that uses the vector registers and functional units of the Cray hardware. The programmer does not need to know the details of the vectorization process because CFT77 automatically vectorizes Fortran programs. When CFT77 cannot vectorize code, it generates scalar code using a variety of optimization techniques to improve execution time. Scalar optimization transforms the internal representation of the Fortran program into a more efficient but functionally equivalent program.

CFT77 is portable on several levels. Because it complies with the 1978 ANSI standard, programs written for other computer systems have maximum portability to the CRAY X-MP EA system with minimum effort. Also, the compiler runs on all Cray systems, so a Fortran program that compiles and runs on one Cray system will run on all Cray systems. Changing from CFT to CFT77 is simple; programs that compile and run correctly with the CFT compiler also do so with CFT77.

CFT

The Fortran compiler (CFT) compiles Fortran programs that make full use of the hardware of the CRAY X-MP EA computer system. CFT automatically generates vectorized machine language code therefore, the power of a vector computer becomes immediately accessible to a user having no prior experience with vectorization techniques. The CFT analyzes the innermost loops of a Fortran program to detect vectorizable sequences. The vectorization of inner loops in Fortran programs allows these programs to take advantage of the high speed of vector operations.

CFT is a highly efficient scalar optimizing compiler with a very high compiling speed. CFT schedules scalar and vector instructions to take full advantage of the multiple independent functional units. CFT can generate re-entrant, stack-based code for use in multitasking applications.

C COMPILER

The C language is a versatile applications and systems programming language. Most of the UNICOS kernel code and utilities are written in C. C's potential has also been realized in programming applications other than operating system code; C offers a large standard library of functions and an ever-expanding base of software application programs. The availability of C complements the scientific orientation of Fortran. The C compiler performs scalar optimization and code vectorizes automatically.

The Cray C compiler is available on all Cray computer systems. The compiler translates C language statements into assembler instructions that make effective use of the appropriate Cray computer system.

The C preprocessor, cpp, is included as a part of the Cray C compiler. Cpp allows macro substitution, conditional compilation, and the inclusion of named files in the compilation process.

PASCAL

Pascal is a high-level, general-purpose programming language used as the implementation language for the CFT77 compiler and other Cray products. Cray Pascal complies with the ISO Level 1 standard and offers such extensions to the standard as separate compilation of modules, imported and exported variables, and an array syntax.

The Pascal compiler transforms Pascal code into machine language instructions that run on Cray computer systems. Using Pascal, a programmer can implement algorithms and data structures in a high-level, machine-independent manner without sacrificing efficiency.

The Cray Pascal compiler takes advantage of CRAY X-MP EA hardware features through scalar optimization and automatic vectorization. The compiler provides access to Fortran common block variables and uses a common calling sequence that allows Pascal code to call Fortran and CAL routines.

CRAY ASSEMBLER

The Cray assembler, CAL, enables a user to closely tailor a program to the architecture of the CRAY X-MP EA computer system. Through CAL, a programmer may symbolically express all hardware functions of the Cray system. CAL allows the production of highly efficient machine language programs. The user may designate program and data information to allow complete control of the mainframe CPUs.

A set of versatile pseudo operations for defining macro instructions and controlling the assembler enhances the basic instruction set. A macro library provides macro instructions for subroutine entry and exit allowing for easy subroutine linkage.

SUBROUTINE LIBRARIES

Cray software includes subroutines that are callable from CFT77, CFT, C, Pascal, and CAL. The subroutines are divided into libraries generally on a functional basis. Libraries containing utility, common mathematical, and I/O subroutines are available, as are special purpose libraries.

UTILITIES

A set of software tools assists both interactive and batch users in the efficient use of the CRAY X-MP EA computer system.

The segment loader SEGLDR, is an automatic loader for code produced by the language processors CFT77, CFT, C, Pascal, and CAL. SEGLDR can also be explicitly controlled by the programmer. Program segments are loaded as required without calls to an overlay manager.

The Symbolic Debugging Package allows users to detect program errors by examining both running programs and program memory dumps. The package includes symbolic interactive debuggers and symbolic postmortem dump interpreters.

Performance aids assist in analyzing program performance and optimizing programs with a minimum of effort.

The UPDATE program offers a facility especially convenient for modifying very large programs and datafiles.

Operational support facilities allow proper management of the system.

I/O SUBSYSTEM SOFTWARE

The I/O Subsystem (IOS) software is written in the I/O Processor (IOP) macro assembly language APML. The APML assembler runs in the CPU under control of the operating system and generates the IOS objective code .

The IOS kernel and associated overlays serve as the IOS operating system. A copy of the kernel runs in each IOP, dynamically adjusting at deadstart to individually assigned configurations and functions. The IOS disk subsystem software supports the disk storage units attached to the IOS. This software emphasizes high I/O performance while minimizing the CPU overhead and runs on the BIOP and, optionally, the DIOP.

The IOS software for the XIOP includes the block multiplexer channel interface and tape support. The block multiplexer channel interface allows communications from the IOS to IBM-compatible devices over IBM protocol channels. A block multiplexer tape processes requests for tape I/O from UNICOS or COS.

Cray software treats the IOS Buffer Memory as a high-performance disk. Buffer Memory is partitioned at IOS deadstart; an area is set aside for IOS use (buffers, overlays, and tables), and the remainder is available to users for data sets or files. Partitioning is controlled by parameters specified in the site's configuration overlay.

COMMUNICATIONS SOFTWARE

The CRAY X-MP EA computer systems fit into environments consisting of single or multiple Cray systems, other vendors' mainframes, minicomputers, workstations, and devices capable of high-speed data transfer. CRI provides easy user access to Cray system capabilities, the ability to distribute applications between Cray computer systems and other vendor systems, and effective integration into existing customer networks.

The Transport Control Protocol/Internet Protocol (TCP/IP) allows the CRAY X-MP EA computer system to function as a peer in TCP/IP-supported, open networking environments. TCP/IP is a set of computer networking protocols that allow two or more hosts to communicate; it also is a set of procedures that allow communication between all hosts on a network whether the systems are similar or not.

The TCP/IP networking protocols were defined by the U.S. Department of Defense and enhanced by the University of California at Berkeley with the UNIX system. TCP/IP is supported under UNICOS; it is not supported under COS.

CRI station software products provide CRAY X-MP EA systems access to proprietary protocol implementations through network gateways. CRI supplies the station software packages for various front-end systems. These packages support batch job submission, job status, job control, file transfer, and interactive access to Cray systems.

The following stations are available:

- The Apollo station provides the software connection between the Cray mainframe and the Apollo DOMAIN workstation network.
- The CYBER station joins the Cray system to the Control Data Corporation CYBER 180 series, 70/170, or 700/800 systems to form a powerful computing combination.
- The VAX/VMS station controls the hardware and software link between a DEC VAX computer system and a Cray computer system.
- The MVS station provides the software connection between an IBM System/370, Extended Architecture, or compatible computer system and a Cray computer system.
- The VM station enables IBM-compatible systems running under control of the Virtual Machine/System Product (VM/SP) and Conversational Monitor System (CMS) to be linked with a Cray computer system.
- The UNIX station provides Cray operating system access to installations whose front ends run UNIX.

The Station software for UNISYS, Honeywell Bull, and Data General AOS operating system is available from third-party vendors.

APPLICATIONS

CRI supports application software vendors in the conversion and optimization of software for the CRAY X-MP EA computer system. Many of the most widely used application programs are currently available and supported to run with UNICOS and COS systems. These codes are in fields such as computational fluid dynamics, structural analysis, mechanical engineering, nuclear safety, circuit design, seismic processing, image processing, molecular modeling, and artificial intelligence.

The availability of applications for Cray systems is determined largely by customer requirements communicated to the software vendors. CRI supports the ongoing process of converting and maintaining application software.

SOFTWARE PUBLICATIONS

A partial list of CRI software publications appears in the following subsection. The manuals provide additional information about the software described in this section. These manuals and other user publications can be ordered through CRI local or regional sales offices. Refer to the *User Publication Catalog* (CRI publication number CP-0099) for a complete list of software publications.

UNICOS Operating System

Publication Number	Title
SG-2010	UNICOS Primer
SR-2011	UNICOS User Commands Reference Manual
SR-2012	UNICOS System Calls Reference Manual
SR-2014	UNICOS File Formats and Special Files Reference Manual
SR-2022	UNICOS Administrator Commands Reference Manual
SG-2005	I/O Subsystem (IOS) Operator's Guide for UNICOS
SG-2017	UNICOS Source Code Control System (SCCS) User's Guide
SG-2018	UNICOS System Administrator's Guide for CRAY Y-MP, CRAY X-MP, and CRAY-1 Computer Systems
SG-2050	UNICOS Text Editor's Primer
SG-2052	UNICOS Overview for Users

COS Operating System

Publication Number	Title
SR-0011	COS Reference Manual
SG-0051	I/O Subsystem (IOS) Operator's Guide for COS

Fortran

Publication Number	Title
SR-0009	Fortran (CFT) Reference Manual
SR-0018	CFT77 Reference Manual
SG-0115	CFT Optimization Guide

C

Publication Number	Title
SR-2024	Cray C Reference Manual
SR-0136	CRAY Y-MP, CRAY X-MP, and CRAY-1 C Library Reference Manual

Pascal

Publication Number	Title
SR-0060	Pascal Reference Manual

Cray Assembler

Publication Number	Title
SR-2003	CAL Assembler Version 2 Reference Manual
SR-0085	CRAY Y-MP and CRAY X-MP Symbolic Machine Instructions Reference Manual

Libraries and Macros

Publication Number	Title
SR-0113	Programmer's Library Reference Manual

Utilities

Publication Number	Title
SR-0066	Segment Loader (SEGLDR) Reference Manual
SR-0112	Symbolic Debugging Package Reference Manual
SR-0222	CRAY Y-MP and CRAY X-MP Multitasking Programmer's Manual
SR-0013	UPDATE Reference Manual

Communications Software

Publication Number	Title
SA-0250	Apollo DOMAIN Station Reference Manual
SC-0270	CDC NOS/VE Link Software Command Reference Manual
SR-0034	CDC NOS/BE Station Reference Manual
SR-0035	CDC NOS Station Reference Manual
SG-2009	TCP/IP Network User's Guide
SI-0038	IBM MVS Station Reference Manual
SI-0160	IBM VM Station Command and Reference
SU-0107	UNIX Station User's Guide
SV-0020	DEC VAX/VMS Station Reference Manual

Applications

Publication Number	Title
ASD-87	Directory of Applications Software for Cray Supercomputers

SOFTWARE TRAINING

CRI offers complete training on the software available for the CRAY X-MP EA computer system. Extensive user-support analyst and system analyst training is available at CRI's training facility. End-user, operators and system administrator training are available at customer sites after installation of a Cray computer system. More information regarding courses and schedules can be obtained through your local or regional CRI sales office.

GLOSSARY

A

A register - Address register. A registers are primarily used as address registers for memory references and as index registers.

B

B register - Intermediate Address register. B registers are used as intermediate storage for the A registers.

BIOP - Buffer I/O Processor. The BIOP, one of the I/O Processors in the I/O Subsystem, transfers data between Central Memory and Buffer Memory.

BM - Bidirectional Memory (flag). The M register in the Exchange Package contains the BM flag. When the BM flag is set, block reads and writes can operate concurrently.

BMC - Block Multiplexer Controller. The BMC provides a hardware interface to IBM and IBM-compatible peripheral controllers and their attached peripheral devices.

Buffer Memory - The shared memory in the IOS common to all I/O processors.

C

CA - Current Address (register). The CA register contains the initial address for a channel transfer. The contents of the CA register are incremented until the transfer is complete.

CAL - Cray Assembly Language. A symbolic language that generates machine instructions on a one-for-one basis and allows programs to call subroutines from the library through the use of macro instructions.

CCI - Clear Clock Interrupt (instruction). The CCI instruction clears a programmable clock interrupt request.

Central Memory - Memory in the CRAY X-MP EA mainframe shared by all CPUs.

CFT - Cray Fortran compiler. CFT is a fast, vectorizing compiler that is fully compliant with the ANSI 78 standards.

CFT77 - Cray Fortran compiler. CFT77 performs a high degree of vector and scalar optimization and is fully compliant with the ANSI 78 standards.

CI - Clear Interrupt flag instructions. The CI instructions clear the channel interrupt.

C (continued)

CIP - Current Instruction Parcel (register). The CIP register holds the instruction waiting to issue.

CIPI - Clear interprocessor interrupt (instruction). The CIPI instruction clears an interprocessor interrupt.

CL - Channel Limit (register.) The contents of the CL register are one greater than the last address for a channel transfer. When the contents of the CA register equals the contents of the CL register, the transfer is complete.

CLN - Cluster Number (register). The CLN register, part of the Exchange Package, determines the CPU's cluster. The CLN register contents are used to determine which set of SB, ST, and SM registers the CPU can access.

CMR - Complete Memory Reference (instruction). The CMR instruction assures completion of all memory references within a particular CPU issuing the instruction.

COS - Cray operating system. COS is a multiprogramming, multiprocessing, and multitasking operating system for Cray mainframes.

CP - Clock period. The CP is the interval in which the system clock completes one oscillation.

CPU - Central processing unit. The CPU is the primary functioning unit of the computer system. It consists of a computation section and a control section.

CRI - Cray Research, Inc.

D

DBA - Data Base Address (register). The DBA register, part of the Exchange Package, holds the base address of the user's data range.

DBM - Disable Bidirectional Memory transfers (instruction). The DBM instruction disables the bidirectional memory mode.

DCI - Disable Clock Interrupts (instruction). The DCI instruction disables programmable clock interrupts until an enable clock interrupt instruction is executed.

DCU - Disk controller unit. The DCU interfaces the disk storage units to an I/O Processor within the I/O Subsystem.

Deadlock - A state resulting in the inability to continue processing due to an unresolvable conflict. Deadlock occurs when all CPUs in a cluster are holding issue on a Test and Set instruction.

Deadstart - The sequence of operations required to start an operating system running in a Cray computer system.

DFI - Disable Floating-point Interrupts (instruction). The DFI instruction clears the Floating-point Interrupt flag in the Mode register.

DIOP - Disk I/O Processor. The DIOP, one of I/O Processors in the I/O Subsystem, performs disk I/O to and from disk storage units attached through controllers to the DIOP's channels.

D (continued)

DL - Deadlock (flag). The F register in the Exchange Package contains the DL flag. If all CPUs in a cluster are holding issue on a Test and Set instruction, the DL flag is set in each CPU in the cluster that is not in monitor mode and an exchange occurs.

DLA - Data Limit Address (register). The DLA register holds the upper limit address of the user's data range.

DRI - Disable Interrupt on Address Range error (instruction). The DRI instruction clears the Operand Range Error Mode flag in the Exchange Package M register.

DSU - Disk Storage Unit is controlled by the channel functions from the I/O processor through the Disk Controller Unit.

E

EAM - Extended Addressing Mode (flag). The M register contains the EAM flag. When the EAM flag is set, it indicates that 32-bit (Y-mode) addressing takes place. When it is not set, it indicates that 24-bit (X-mode) addressing takes place.

EBM - Enable Bidirectional Memory transfers (instruction). The EBM instruction enables the bidirectional memory mode.

ECI - Enable Clock Interrupts (instruction). The ECI instruction enables programmable clock interrupts at a rate determined by the value in the Interrupt Interval register.

EEX - Error Exit (flag). The F register in the Exchange Package contains the EEX flag. The EEX flag sets when an Error Exit instruction executes and monitor mode is not in effect.

EFI - Enable Floating-point Interrupt (instruction). The EFI instruction sets the Floating-point Interrupt flag in the Mode register.

ERI - Enable interrupt an Address Range error (instruction). The ERI instruction sets the Operand Range Error flag in the exchange package M register.

ERR - Error Exit (instruction). The ERR instruction initiates an exchange sequence. If monitor mode is not in effect, the instruction sets the EEX flag.

ERR - Read Error Type (bit). The type of memory error encountered, correctable or uncorrectable, is indicated in this bit of the Exchange Package. The error type is one of the memory error data fields in the Exchange Package.

ESVL - Enable Second Vector Logical (bit). The contents of the ESVL bit in the Exchange Package indicate whether or not the Second Vector Logical functional unit can be used.

EX - Normal Exit (instruction). The EX instruction initiates an exchange sequence.

Exchange Mechanism - The technique used in the CRAY X-MP EA computer system for switching instruction execution from program to program. Refer to Exchange Package.

E (continued)

Exchange Package - A 16 word block of data in memory reserved for exchange packages. The Exchange Package contains the necessary registers and flags associated with a particular program. Each program has its own Exchange Package.

Exchange sequence - Moving an inactive Exchange Package from memory into the operating registers and at the same time moving the currently active Exchange Package from the operating registers back into memory.

F

F register - Flag register. The F register contains part of the Exchange Package for the currently active program. The F register contains flags identified individually within the Exchange Package. Setting any of these flags interrupts program execution.

F - Floating-point (operation). When an F appears in front of a register designator in a symbolic machine instruction, the calculation is a floating-point operation.

FEI-1 - Front-end Interface. An interface that connects the CRAY X-MP EA computer I/O channels to channels of front-end computers. An FEI compensates for differences in channel widths, machine word size, electrical logic levels, and control signals.

FEI-3 - Front-end Interface. A group of front-end interfaces that enable certain VME-based microcomputers and workstations to communicate with a Cray system over a standard 6-Mbyte/s I/O channel.

FOL - Fiber-optic link. The CRI fiber-optic link allows an FEI-1 to be separated from a Cray computer system by distances of up to 3,281 ft (1,000 m). The FOL provides complete electrical separation of the connected devices.

FPE - Floating-point Error (flag). The F register in the Exchange Package contains the FPE flag. The FPE flag sets when a Floating-point Range error occurs in any of the floating-point functional units and the Enable Floating-point Interrupt flag is set.

FPS - Floating-point Error Status (flag). The M register in the Exchange Package contains the FPS flag. When the FPS flag is set, a Floating-point error occurred regardless of the state of the Interrupt on Floating-point Error flag.

Functional Unit - Hardware within a CRAY X-MP EA mainframe that performs specialized functions. Functional units perform arithmetic, logical, shift and other functions. All functional units can operate concurrently.

G

GOS - Guest Operating System. GOS runs under COS and allows UNICOS to run concurrently with COS in the same mainframe.

H

H - Half-precision floating-point (operation). When an H appears in front of a register designator in a symbolic machine instruction, the calculation is a half-precision floating-point operation.

HSX - High-speed External (HSX) Communications channel. The HSX channel is a high-speed, full-duplex, external channel capable of transferring data at a maximum rate of 100 Mbytes/s.

I

I - Reciprocal iteration (operation). When an I appears in front of a register designator in a symbolic machine instruction, the calculation is a reciprocal iteration operation.

IBA - Instruction Base Address (register). The IBA register is in the Exchange Package. The IBA register holds the base address of the user's instruction range.

ICM - Correctable Memory Error (flag). The M register in the Exchange Package contains the ICM flag. When the ICM flag is set, it enables interrupts on correctable memory data errors.

ICP - Interrupt from Internal CPU (flag). The F register in the Exchange Package contains the ICP flag. The ICP flag sets when another CPU issues a SIPI instruction.

IFP - Floating-point Error Mode (flag). The M register in the Exchange Package contains the ICM flag. When the IFP flag is set, it enables interrupts on floating-point errors.

ILA - Instruction Limit Address (register). The ILA register is in the Exchange Package. The ILA register holds the limit address of the user's instruction field.

IMM - Interrupt Monitor Mode (flag). The M register in the Exchange Package contains the IMM flag. When the IMM flag is set, it enables all interrupts in monitor mode except DL, PCI, MCU, NEX, and IOI.

Instruction buffer - A set of registers in a CRAY X-MP EA mainframe used for temporary storage of instructions before issue. Each Instruction buffer can hold 128 consecutive instruction parcels.

Instruction fetch - The process of loading program code from Central Memory to an Instruction buffer.

IOI - I/O Interrupt (flag). The F register in the Exchange Package contains the IOI flag. The IOI flag sets when a 6-Mbyte/s channel or the 1,000-Mbyte/s channel (100 Mbytes/s channel in single-processor models) completes a transfer.

IOP - I/O Processor. An IOP is a fast, multipurpose computer capable of transferring data at extremely high rates. Multiple IOPs are housed in an I/O Subsystem.

IOR - Operand Range Error (flag). The M register in the Exchange Package contains the IOR flag. When the IOR flag is set, it enables interrupts on operand address range errors.

IOS - I/O Subsystem. The IOS provides high-capacity data communications between Central Memory of a Cray mainframe and peripheral devices, data storage devices, and front-end computers.

Issue - The process of reading an instruction from an instruction buffer and starting its execution.

I (continued)

IUM - Uncorrectable Memory Error (flag). The M register in the Exchange Package contains the IUM flag. When the IUM flag is set, it enables interrupts on uncorrectable memory data errors.

J

J - The unconditional branch instruction.

JAM - A conditional branch instruction. A branch occurs if the contents of register A0 is negative.

JAN - A conditional branch instruction. A branch occurs if the contents of register A0 is nonzero.

JAP - A conditional branch instruction. A branch occurs if the contents of register A0 is positive.

JAZ - A conditional branch instruction. A branch occurs if the contents of register A0 is 0.

JSM - A conditional branch instruction. A branch occurs if the contents of register S0 is negative.

JSN - A conditional branch instruction. A branch occurs if the contents of register S0 is nonzero.

JSP - A conditional branch instruction. A branch occurs if the contents of register S0 is positive or 0.

JSZ - A conditional branch instruction. A branch occurs if the contents of register S0 is 0.

L

LIP - Lower Instruction Parcel (register). The LIP register holds the second parcel of a 2-parcel or 3-parcel instruction prior to issue.

LIP-1 - Lower Instruction Parcel (register). The LIP-1 register holds the third parcel of a 3-parcel instruction prior to issue.

M

M register - Mode register. The M register in the Exchange Package contains user-selectable flags that dictate the execution of the program.

MCU - Maintenance Control Unit Interrupt (flag). The F register in the Exchange Package contains the MCU flag. The MCU flag sets when the MIOP sends this signal.

ME - Memory Error (flag). The F register in the Exchange Package contains the ME flag. The ME flag sets when a correctable or uncorrectable memory error occurs and the corresponding Interrupt on Memory Error bit is set in the M register.

MG - Motor Generator. An MG converts commercial electrical power to the voltage and frequency required by the CRAY X-MP EA computer system and isolates the computer system from power line variations.

MIOP - Master I/O Processor. The MIOP, one of the I/O Processors in the I/O Subsystem, initializes the contents of Buffer Memory and deadstarts the other processors.

M (continued)

MM - Monitor Mode (flag). The M register in the Exchange Package contains the MM flag. When the MM flag is set, it inhibits all interrupts except memory errors, normal exit, and error exit.

Mode - Read mode (bits). The Mode bits are part of the memory error data fields in the Exchange Package. The Mode bits determine the type of read mode in progress when a Memory Data error occurred; these bits are used with the port bits to identify the operation in error.

Monitor Mode - A condition in which a CPU inhibits all interrupts except those caused by memory errors or normal or error exit instructions.

N

NEX - Normal Exit (flag). The F register in the Exchange Package contains the NEX flag. The NEX flag is set by a normal exit instruction if not in monitor mode or if the Interrupt Monitor Mode bit is not set.

NIP - Next Instruction Parcel (register). The NIP register holds a parcel of program code before it enters the Current Instruction Parcel register. Instruction decoding begins in this register.

O

ORE - Operand Range Error (flag). The F register in the Exchange Package contains the ORE flag. The ORE flag sets when a data reference is made outside the boundaries of the DBA and DLA registers and the Interrupt on Operand Range Error flag is set.

P

P - Population count (operation). When a P appears in front of a register designator in a symbolic machine instruction, the calculation is a population count operation.

Parcel - A 16-bit portion of a word that is addressable for instruction execution but not for operand references.

P register - Program Address register. The P register selects an instruction parcel from one of the instruction buffers. The contents of the P register are stored in the Program Address register field in the Exchange Package. The instruction at this location is the first instruction issued when this program begins.

PCI - Programmable Clock Interrupt (flag). The F register in the Exchange Package contains the PCI flag. The PCI flag sets when the interrupt countdown counter in the programmable clock equals 0.

PDU - Power Distribution Unit. The PDU contains adjustable transformers for regulating the voltage to each piece of equipment. It also contains the temperature and voltage monitoring equipment that checks temperatures at strategic locations. The CRAY X-MP EA mainframe, IOS, and SSD require PDUs.

PN - Processor Number. The PN field in an Exchange Package indicates which CPU executed the exchange sequence.

P (continued)

PRE - Program Range Error (flag). The F register in the Exchange Package contains the PRE flag. The PRE flag sets when an instruction fetch is made outside the boundaries of the IBA and ILA registers.

Programmable Clock - A 32-bit counter in each CPU that is used to generate interrupts at selectable intervals.

PS - Program State (flag). The M register in the Exchange Package contains the PS flag. The PS flag is set by the operating system to show whether a CPU concurrently processing a program with another CPU is the master or slave.

Q

Q - Parity count (operation). When a Q appears in front of a register designator in a symbolic machine instruction, the calculation is a parity count operation.

R

R - Rounded floating-point (operation). When an R appears in front of a register designator in a symbolic machine instruction, the calculation is a rounded floating-point operation.

RCU -Refrigeration Condensing Unit. The RCU transfers heat from a refrigerant gas to an external water supply, causing the gas to condense.

RT - Load Real-time Clock (instruction). The RT instruction loads the Real-time Clock register with the contents of an S register.

RTC - Real-time Clock. The RTC is a 64-bit counter that advances one count each clock period.

S

S register - Scalar register. The S registers are the source and destination registers for operands executing scalar arithmetic and logical instructions.

SB - Shared Address register. The SB register is a shared register used for passing address information from one CPU to another.

SB - Sign Bit. SB is the CAL syntax used in the machine instruction 050*ij*0 to scalar merge the contents of the S*i* register and the sign bit of the S*j* register into the S*i* register.

SECDED - Single-error correction/double-error detection. SECDED assures that data written into Central Memory is read with consistent precision. If a single bit of a data word is altered, alteration is automatically corrected when the word is read from memory. If 2 bits of the same data word are altered, the error is detected but cannot be corrected.

SEI - Selected for External Interrupts (flag). The M register in the Exchange Package contains the SEI flag. When the SEI flag is set, this CPU is preferred for I/O interrupts.

S (continued)

SIPI - Set interprocessor interrupt (instruction). The SIPI instruction sets an interprocessor interrupt request to a specific CPU.

SM - Semaphore (register). The SM register is a shared register used for control between CPUs.

SSD - SSD solid-state storage device. The SSD is a high-performance device used for temporary data storage.

ST register - Shared Scalar register. The ST register is a shared register used for passing scalar information from one CPU to another.

Status register - A read-only register that is used to determine the operating modes of a CPU.

T

T register - Intermediate Scalar register. The T registers are used as intermediate storage for the S registers.

U

UNICOS - The UNICOS operating system is derived from the AT&T UNIX System V operating system. UNICOS is also based in part on the Forth Berkely Software Distribution under license from The Regents of the University of California. UNICOS is essentially the same in philosophy, structure, and function as UNIX but has been enhanced to exploit the power of Cray computer systems.

V

V register - Vector register. Each V register contains sixty-four 64-bit elements.

VL - Vector Length (register). The program-selectable VM register controls the effective length of a vector register for any operation.

VM - Vector Mask (register). The VL register allows for the logical selection of particular elements of a vector.

VNU - Vector Not Used (bit). The state of the VNU bit in the Exchange Package indicates whether several specific vector instructions were issued during the program execution interval.

W

WS - Waiting for Semaphore (flag). The M register in the Exchange Package contains the WS flag. When the WS flag is set, the CPU exchanged when a Test and Set instruction was holding in the CIP register.

X

XA - Exchange Address (register). The XA register in the Exchange Package specifies the first word address of a 16-word Exchange Package loaded by an exchange operation.

X-mode - The X-mode is selected by setting the Extended Addressing Mode bit in the Exchange Package to 0. When the mainframe is operating in the X-mode, the A registers, B registers, address functional units, and address memory references are limited to 24 bits. Only 1-parcel and 2-parcel instructions run.

XIOP - Auxiliary I/O Processor. The XIOP, one of the I/O Processors in the I/O Subsystem, interfaces to BMC-5 block multiplexer controllers. It manages the data from IBM-compatible tape drives and buffers the data to Buffer Memory.

Y

Y-mode - The Y-mode is selected by setting the EAM bit in the Exchange Package to 1. When the mainframe is operating in the Y-mode, the A registers, B registers, address functional units, and address memory references run at full 32-bit width. The instruction set includes 1-parcel, 2-parcel, and 3-parcel instructions.

Z

Z - Leading-zero count (operation). When a Z appears in front of a register designator in a symbolic machine instruction, the calculation is a leading-zero count operation.

INDEX

READER COMMENT FORM

CRAY X-MP Extended Architecture Computer Systems Functional Description Manual HR-3020

Your comments help us to improve the quality and usefulness of our publications. Please use the space provided below to share with us your comments. When possible, please give specific page and paragraph references.

NAME ____________________

JOB TITLE ____________________

FIRM ____________________

ADDRESS ____________________

CITY __________ STATE ______ ZIP ______

DATE ____________________

CRAY RESEARCH, INC.

CUT ALONG THIS LINE

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY
RESEARCH, INC.

Attention: PUBLICATIONS
890 Industrial Boulevard
Technical Operations Building
Chippewa Falls, WI 54729

STAPLE