CRAY Y-MP C90
Functional Description Manual

HR-04028

Cray Research, Inc.

CRAY Y-MP C90
Functional Description Manual

HR-04028

Cray Research, Inc.

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

CRAY RESEARCH, INC.
Distribution
2360 Pilot Knob Road
Mendota Heights, MN 55120
(800) 284-2729 extension 5907

Comments about this publication should be directed to:

CRAY RESEARCH, INC.
Hardware Publications and Training
770 Industrial Blvd.
Chippewa Falls, WI 54729

Reader Comment Form

Title: CRAY Y-MP C90 Functional Description Manual **Number: HR-04028**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this manual?

_____Troubleshooting
_____Tutorial or introduction
_____Reference information
_____Classroom use
_____Other - please explain _____

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one)

_____Too much information _____
_____Too little information _____
_____Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME _____
JOB TITLE_____
FIRM_____
ADDRESS_____
CITY_____STATE_____ZIP_____
DATE_____

[or attach your business card]

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.

Attn: Hardware Publications and Training
770 Industrial Boulevard
Chippewa Falls, WI 54729

STAPLE

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
	March 1992. Original printing.

PREFACE

The CRAY Y-MP C90 functional description manual describes the basic functions, design, and architecture of the CRAY Y-MP C90 computer system and its associated peripheral devices. This manual is written primarily for Cray Research customers.

This manual is divided into the following tabbed sections.

Section 1, "System Overview," introduces and describes the CRAY Y-MP C90 system components and support equipment.

Section 2, "Mainframe," describes the basic hardware architecture and CPU instructions of the CRAY Y-MP C90 mainframe. A specification sheet is included at the end of this section.

Section 3, "I/O Subsystem," describes the basic architecture and functions of the input/output subsystem (IOS). A specification sheet is included at the end of this section.

Section 4, "SSD Solid-state Storage Device," describes the basic architecture of the SSD solid-state storage device model E. A specification sheet is included at the end of this section.

Section 5, "Peripheral Equipment," describes the function of the disk drives and network interface equipment used by the CRAY Y-MP C90 computer system. Specification sheets are included at the end of this section.

Section 6, "Software Overview," provides an overview of the software available for the CRAY Y-MP C90 computer system.

For the readers' convenience, a glossary is included. It defines many of the commonly used abbreviations and terminology associated with the CRAY Y-MP C90 computer system.

The following conventions are used throughout this manual.

Convention	Description
Lowercase italic	Variable information.
X or x or *x*	An unused value.
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. One exception is 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. Another exception is when the contents of the 32 1-bit semaphore registers are loaded into an S register. SM0 goes into S register bit position 2^{63}, SM1 goes into S register bit position 2^{62}, and so on.
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 list provides 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 *x* indicates that 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.

CONTENTS

1 SYSTEM OVERVIEW

2 MAINFRAME

2 MAINFRAME (continued)

3 I/O SUBSYSTEM

4 SSD SOLID-STATE STORAGE DEVICE

5 PERIPHERAL EQUIPMENT

5 PERIPHERAL EQUIPMENT (continued)

6 SOFTWARE OVERVIEW

FIGURES

FIGURES (continued)

TABLES

GLOSSARY

BIBLIOGRAPHY

INDEX

1 SYSTEM OVERVIEW

The CRAY Y-MP C90 computer system is a powerful, general-purpose supercomputer. The large memory, fast clock speed, and powerful input/output (I/O) capabilities enable fast throughput, resulting in efficient use of supercomputing power. The CRAY Y-MP C90 computer system achieves extremely high multiprocessing rates by efficiently using the scalar and vector processing capabilities of the multiple central processing units (CPUs), combined with the system's random-access memory (RAM) and shared registers.

A standard CRAY Y-MP C90 computer system consists of the following components:

- A mainframe
- An input/output subsystem model E (IOS-E)
- An optional SSD solid-state storage device model E (SSD-E)
- Mass storage devices such as disk and tape drives
- A maintenance workstation model E (MWS-E)
- An operator workstation model E (OWS-E)
- Network interfaces
- Power and cooling support equipment

The following subsections introduce the system components. Figure 1-1 shows the minimum configuration for a CRAY Y-MP C90 computer system with two I/O clusters. Refer to Figure 1-1 when reading the following subsections.

Subsequent tabbed sections provide more detailed information on the mainframe, IOS-E, SSD-E, peripheral devices, and system software.

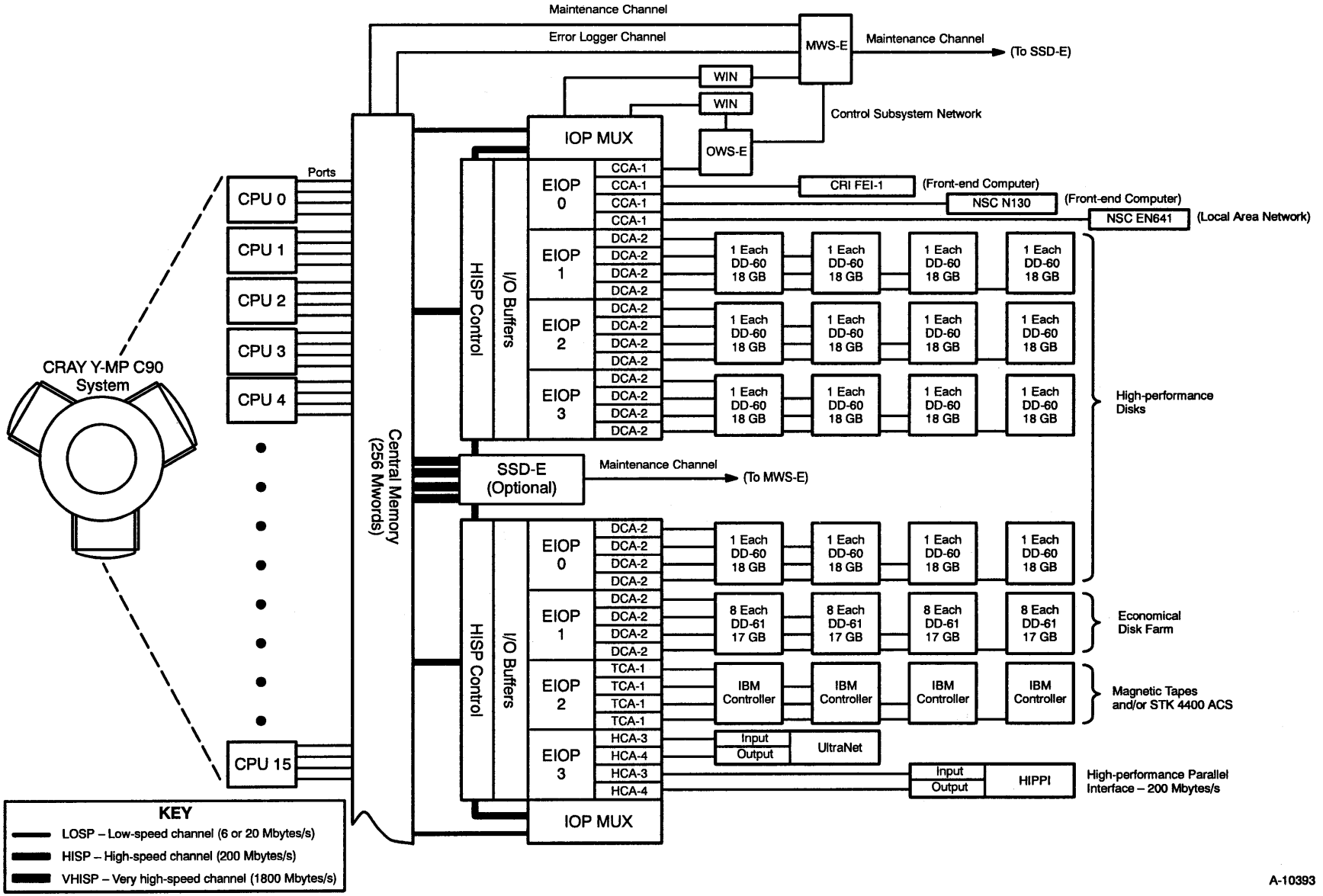


Figure 1-1. Minimum Configuration for CRAY Y-MP C90 Computer System with Two I/O Clusters

Mainframe

A CRAY Y-MP C90 mainframe contains an I/O section, an interprocessor communication section, central memory, and a variable number of CPUs. All CPUs in multiprocessor systems share the I/O section, interprocessor communication section, and central memory.

The I/O section provides high-speed data transfers to and from the IOS-E and SSD-E. The I/O section contains three types of I/O channels:

- 6- or 20-Mbyte/s LOSP channels
- 200-Mbyte/s HISP channels
- 1800-Mbyte/s VHISP channels

The 6- or 20-Mbyte/s channels carry control information between the mainframe and IOS-E. The 200-Mbyte/s channels carry data between the mainframe and IOS-E. The 1800-Mbyte/s channels carry data between the mainframe and the SSD-E. The quantity of each channel type varies with different system configurations and depends on the quantity of CPUs and I/O clusters. Basically, each of the CPUs in the mainframe is configured with one LOSP channel and either two HISP channels or one VHISP channel.

The interprocessor communication section enables each mainframe CPU to synchronize operation and pass data with other CPUs. Central memory holds program code and data. Central memory is available in different sizes and configurations.

Each CPU has a control section and a computation section. The control section determines instruction issue and use of the computation section, central memory, and I/O resources. The computation section consists of operating registers and functional units.

The mainframe is designed to deliver optimum overall performance. Separate registers and functional units support both integer and floating-point computations in both vector and scalar processing modes.

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

The start-up time for vector operations is short enough that vector processing is more efficient than scalar processing for vectors containing as few as two elements. This feature allows fast vector processing to be balanced with high-speed scalar processing.

Multiple-processor systems allow the use of multiprocessing or multitasking techniques. Multiprocessing allows several programs to be 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.

Refer to Section 2, "Mainframe," for more information on the internal operation of the mainframe.

IOS-E/SSD-E

The I/O subsystem model E (IOS-E) and the SSD solid-state storage device model E (SSD-E) are contained in the same cabinet. All CRAY Y-MP C90 computer systems include an IOS-E. The SSD-E, however, is optional and may not be included with your system. The functions of these two system components are explained in the following subsections.

IOS-E

The IOS-E performs the following functions:

- It controls all data transfers between the mainframe or optional SSD-E and peripheral devices such as disk drives and front-end computers.
- It buffers all data transfers between the mainframe or SSD-E and peripheral devices.
- It converts data to and from the formats used by peripheral devices.
- It detects and corrects certain types of data errors that occur during transfers.

The transfer rates between the IOS-E and other equipment vary. Data transfers between the IOS-E and either the mainframe or the SSD-E use 200-Mbyte/s channels. The mainframe and IOS-E transfer control information using the 6- or 20-Mbyte/s channels. The transfer rate between the IOS-E and peripheral devices depends on the peripheral device.

The IOS-E consists of a variable number of I/O clusters and two workstation interfaces (WINs). Each I/O cluster contains one I/O processor multiplexer (IOP MUX), four auxiliary I/O processors (EIOPs), and up to 16 channel adapters. The IOP MUX controls data transfers between the IOS-E and mainframe or SSD-E. The four EIOPs control data transfers between the IOS-E and peripheral devices through the channel adapters. Each EIOP can support a maximum of four channel adapters. The number of clusters and channel adapters varies, depending on the number and types of peripheral devices in the system.

Workstation interfaces allow an OWS-E and MWS-E to control and monitor the operation of the I/O clusters. Refer to Section 3, "I/O Subsystem," for more information on the internal operation of the IOS-E.

SSD-E

The SSD-E is an optional high-performance device used for temporary data storage. It is housed in the same cabinet as the IOS-E. The SSD-E transfers data between the mainframe's central memory and the SSD-E through 1800-Mbyte/s channels. The 1800-Mbyte/s channel operates under mainframe program control. The SSD-E can also connect to the IOS-E by means of a 200-Mbyte/s channel pair. The 200-Mbyte/s channels operate under IOS-E program control. Refer to Section 4, "SSD Solid-state Storage Device," for more information on the internal operation of the SSD-E.

Disk Storage Units

The CRAY Y-MP C90 computer system uses Cray Research disk drives for mass data storage. A disk channel adapter (DCA-1 or DCA-2) interfaces the disk drives to an EIOP. The disk channel adapter can transfer data between the EIOP and multiple disk drives at full speed, even when all the drives are operating simultaneously. Refer to Section 3 for more information on the channel adapters. Refer to Section 5 for more information about the disk storage units.

Tape Drives and Controllers

The IOS-E provides an interface to tape drives and controllers. (Cray Research does not sell tape drives or controllers.) The TCA-1 channel adapter in the IOS-E connects to IBM compatible magnetic tape drives and controllers. Refer to Section 3, "I/O Subsystem," for more information on the channel adapters.

Operator and Maintenance Workstations

The MWS-E and OWS-E (Figure 1-2) are based on a Sun 4/370 SPARCstation, 12-slot chassis. The SPARC (Scalable Processor ARChitecture) is a Sun version of the reduced instruction set computer (RISC) architecture. A VMEbus is provided in slots 4 through 12 of the workstations.

Both workstations run the SunOS 4.1.1 operating system and OpenWindows 2.0 software; the OWS-E also runs the OWS-E software release. The Sun operating system is an enhanced version of UNIX; it combines features of UNIX System Laboratories, Inc.'s System V UNIX and Berkeley Software Distribution's version 4.3 UNIX.

The OWS-E is part of the Cray Research computer system. The MWS-E is owned by Cray Research; it enables Cray Research engineers to perform system maintenance independently of any customer activity on the Cray Research computer system.

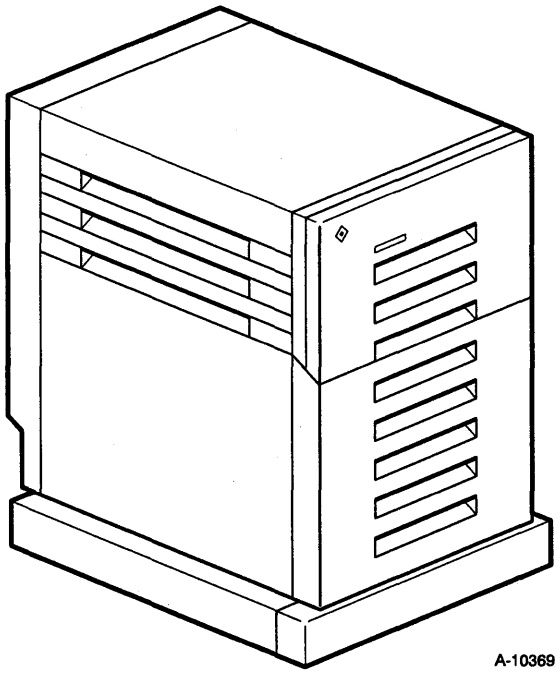


Figure 1-2. MWS-E and OWS-E Workstation Chassis

MWS-E Functions

The MWS-E provides multiple connections for hardware maintenance and monitoring of the CRAY Y-MP C90 computer system. The MWS-E supports Cray Research diagnostics, enhanced diagnostic displays, code simulation, and maintenance and error channels. It monitors

environmental conditions and can shut down the system if severe variances occur. The MWS-E also serves as a platform for remote support, with customer approval.

OWS-E Functions

The OWS-E provides a dedicated workstation that Cray Research analysts and customer operators use to operate, administer, and monitor a Cray Research computer system. The OWS-E is also used for system boot, dump, and clear operations and for software support and upgrades.

Network Interfaces

The CRAY Y-MP C90 computer system is designed to communicate easily and efficiently with front-end computer systems and computer networks.

Standard front-end interfaces (FEIs) connect the I/O channels of the IOS-E to front-end computer channels. These connections provide input data to the system and receive output from the system 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 a stand-alone cabinet located near the host computer; others are installed directly into the front-end computer system. Operation of the FEI is transparent to both the front-end computer users and Cray Research system users.

An optional fiber-optic link (FOL-3 or FOL-4) is available for some FEIs to provide equipment separation distances of up to 13,120 ft (4,000 m). The FOL is installed between the IOS-E and FEI and provides complete electrical separation from the CRAY Y-MP C90 computer system.

Refer to "Network Interfaces" in Section 5 for more information on network interfaces.

Power and Cooling Support Equipment

The logic modules that comprise the mainframe, IOS-E, and SSD-E require special equipment to supply electrical power and to remove heat. The following subsections define the power and cooling support equipment and the warning and control system (WACS) used with CRAY Y-MP C90 computer systems. Refer to Section 5, "Peripheral Equipment," for power and cooling requirements for peripheral devices.

Power Equipment

Motor-generator sets (MGSs) and power supplies provide electrical power to the logic modules and are housed in a stand-alone cabinet. The MGS is typically located in a separate power equipment room. The power supplies are housed in the mainframe cabinet.

An MGS uses power from commercial power mains to generate the proper voltage and frequency used by the mainframe power supplies. Customers must supply one of the following commercial power sources to the MGSs:

- 460 Vac, 3 phase, 60 Hz or
- 398 Vac, 3 phase, 50 Hz

The MGSs supply 208-Vac, 400-Hz power to the power supplies. The MGSs also isolate the system from transients and fluctuations from the commercial power mains. The number of required MGSs varies among systems. Depending on the system configuration, a CRAY Y-MP C90 computer system requires one or two MGSs. An optional standby MGS is available.

The power supplies are housed in the mainframe chassis and IOS-E/SSD-E cabinet. The power supplies convert the 208-Vac, 400-Hz voltage to the necessary DC voltages required for the logic modules. The number and types of power supplies are not optional for the customer.

Cooling Equipment

The CRAY Y-MP C90 computer system uses two models of heat exchanger units (HEUs): HEU-C90 and HEU-E/S. The mainframe is connected to the HEU-C90, which has two pumps and two heat exchange subassemblies. The IOS-E/SSD-E cabinet is connected to the HEU-E/S, which contains a single pump and heat exchanger subassembly. The HEUs transfer heat between the dielectric coolant and the refrigerant.

The CRAY Y-MP C90 computer system uses refrigeration condensing units RCU-5 and RCU-6. The RCUs dissipate the heat transferred from the HEUs.

NOTE: The RCU-5, RCU-6, and MGS-6 are configured with the majority of CRAY Y-MP C90 computer systems. However, some CRAY Y-MP C90 computer systems use different support equipment.

An HEU and RCU, along with customer-supplied chilled water, cool the CRAY Y-MP C90 computer system. The RCU is typically located in a separate equipment power room.

Figure 1-3 is a simplified diagram of the cooling system for the CRAY Y-MP C90 computer system. Cooling is accomplished by three systems: one or two closed-loop dielectric-coolant systems, one closed-loop refrigerant system, and a customer-supplied chilled water system.

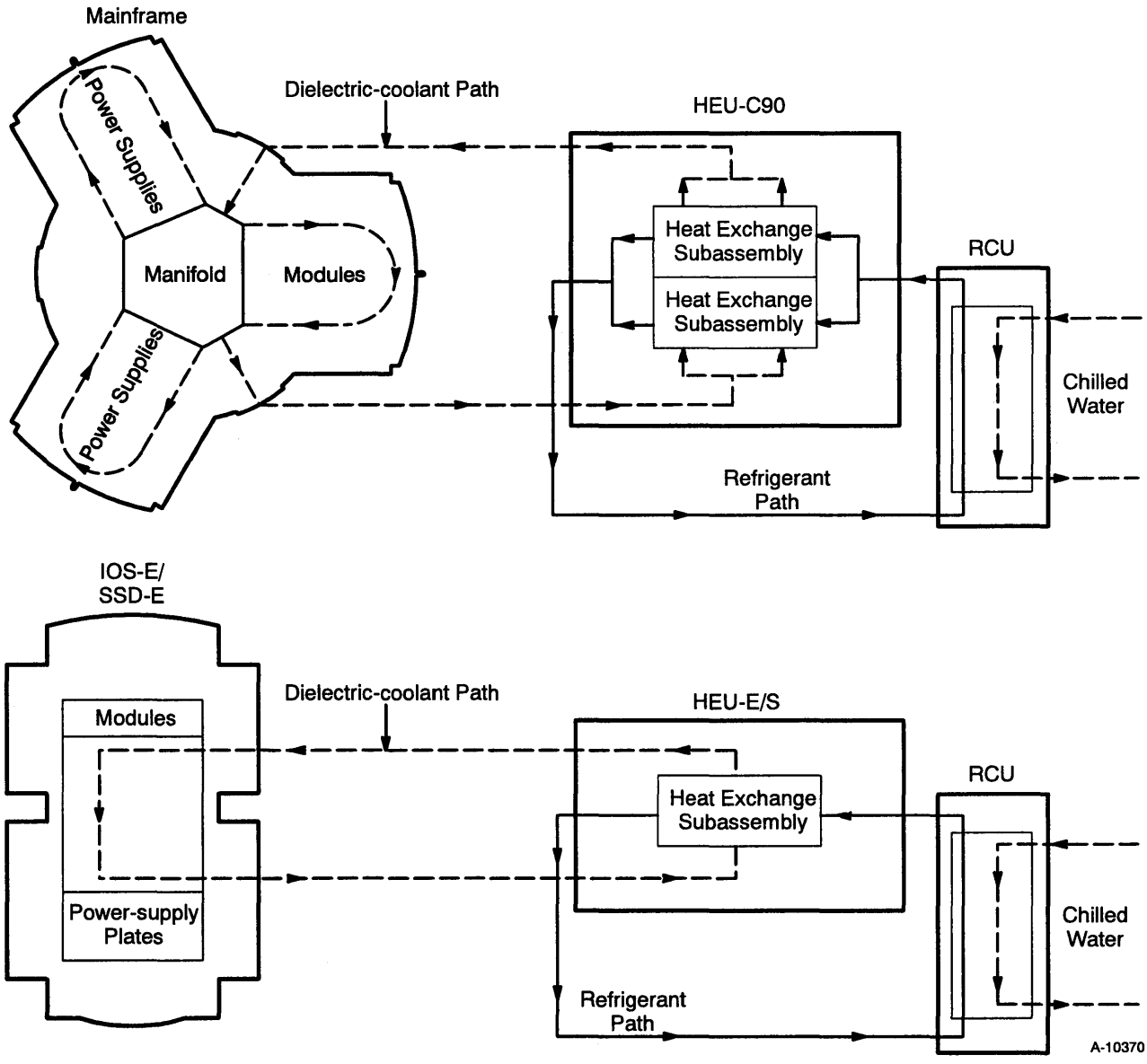


Figure 1-3. CRAY Y-MP C90 Cooling System Block Diagram

Each closed-loop dielectric-coolant system contains a pump that circulates chilled dielectric coolant (such as Fluorinert Liquid) through each module and power-supply mounting plate. The dielectric coolant absorbs heat generated by the modules and power supplies. It then flows to a heat exchange subassembly, where heat transfers from the dielectric coolant to the closed-loop refrigerant system.

WARNING

Prevent dielectric coolant from coming in contact with excessive heat. Dielectric-coolant liquid can decompose and produce hazardous by-products when exposed to excessive heat. Avoid inhaling vapors from Fluorinert Liquid that may have been overheated or exposed to an open flame or electrical arcing. If toxic vapors are inhaled, move the victim to fresh air and call for medical assistance.

The manual *Safe Use and Handling of Fluorinert Liquids*, Cray Research publication number HR-0306, provides specific guidelines and information regarding Fluorinert Liquid.

The closed-loop refrigerant system contains a compressor that circulates the refrigerant. As previously mentioned, the refrigerant absorbs heat from the dielectric coolant. The refrigerant is then circulated through a condenser, where heat transfers to customer-supplied chilled water.

Cray Research recommends a water-supply temperature of approximately 50 °F (10 °C). Other chilled water specifications, such as flow rate and pressure-drop values, vary with different system configurations and actual water-supply temperatures. Cray Research provides additional information on these specifications during the site planning process.

Warning and Control System

The mainframe chassis and the IOS-E/SSD-E cabinet each contain a warning and control system (WACS). The primary function of the WACS is to protect the equipment and environment from damage by continuously monitoring conditions such as temperature and pressure within the cabinet. The WACS can remove electrical power from the cabinet if warning or fault conditions exist. The secondary function of the WACS is to report the warning and fault conditions to the MWS-E.

The WACS consists of printed circuit boards and a system control panel. The WACS monitors the following conditions:

- DC voltages – provides voltage-level monitoring, loss-of-voltage protection, and over-voltage protection
- AC voltages – monitors phase-to-phase and phase-to-neutral voltages
- Temperatures – monitors module and coolant temperatures to protect the mainframe from overheating
- Pressure – provides protection against high- or low-coolant pressure in the cooling system
- Coolant flow – monitors dielectric-coolant flow to protect the cabinet from overheating

The WACS operates on a 120- or 220-Vac, 60-Hz power source, or a 100- or 220-Vac, 50-Hz power source. This power source is separate from the MGS power source.

2 MAINFRAME

This section describes the major functional areas and special features of the CRAY Y-MP C90 mainframe and provides a summary of the Cray Assembly Language (CAL) instruction set. A CRAY Y-MP C90 mainframe specification sheet is included at the end of this section.

CPU Shared Resources

All central processing units (CPUs) in the CRAY Y-MP C90 mainframe share the following resources (refer to Figure 2-1):

- Central memory
- I/O section
- Interprocessor communication section
- Real-time clock

Central Memory

Central memory consists of random-access memory (RAM) that is shared by all the CPUs and the I/O section. Each memory word consists of 80 bits: 64 data bits and 16 error-correcting bits (check bits). Storage for data and check bits is provided by bipolar complementary metal oxide semiconductor (BiCMOS) chips. In order to improve memory access speed, central memory is divided into multiple banks that can be active simultaneously.

In each CPU, the operating registers, instruction buffers, and exchange package have access to central memory through memory ports. Each CPU has four ports. Each of these ports is 2 words wide, allowing up to eight simultaneous memory references from each CPU. The I/O section shares one port in each CPU.

The CRAY Y-MP C90 mainframe central memory uses a single-byte error correcting/double-byte error detecting (SBCDBD) memory error-correcting scheme instead of the single-error correction/ double-error detection (SECDED †) method used in previous Cray Research machines. SBCDBD ensures that data written into central memory is read with consistent precision. If a single byte (4 bits) of data

† Hamming, R. W. "Error Detection and Correcting Codes." *Bell System Technical Journal*. 29.2 (1950): 147–160.

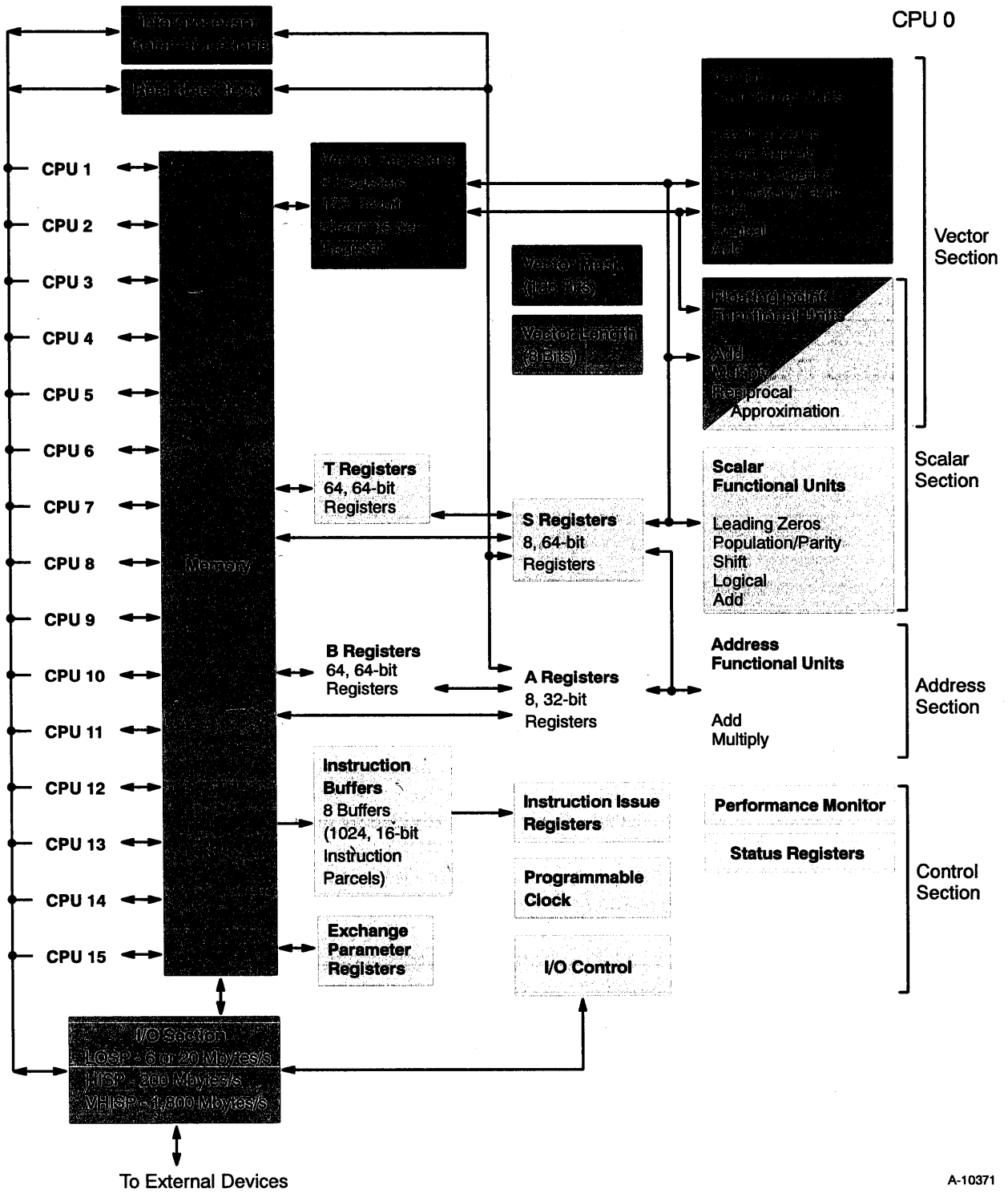


Figure 2-1. CRAY Y-MP C90 Mainframe Block Diagram

is corrupted, the byte is automatically corrected when the word is read from memory. If 2 or more bytes are corrupted, then a double-byte error has occurred and can be detected, but not corrected.

I/O Section

All CPUs share the I/O section of the computer system. The computer system supports three channel types, which are identified by their maximum transfer rates:

- Low-speed (LOSP) channels - 6 or 20 Mbytes/s
- High-speed (HISP) channels - 200 Mbytes/s
- Very high-speed (VHISP) channels - 1,800 Mbytes/s

Interprocessor Communication Section

The interprocessor communication section of the computer system contains shared and semaphore registers to pass data and control information between CPUs. It also contains logic to enable any CPU in monitor mode to interrupt any other CPU and cause it to switch from user to monitor mode. These features are especially useful in multitasking environments.

The shared and semaphore registers are divided into identical groups called clusters. Each cluster contains eight 32-bit shared address (SB) registers, eight 64-bit shared scalar (ST) registers, and thirty-two 1-bit semaphore (SM) registers. Each CPU is assigned to one cluster, giving it access to the registers in that cluster.

The shared registers function as intermediate storage between CPUs and provide a way to transfer data between operating registers in different CPUs. One CPU loads a shared register from its address or scalar registers; other CPUs assigned to the same cluster can then transfer the data from the shared register to their own address or scalar registers. Within a CPU, data is transmitted between the SB and address registers and between the ST and scalar registers.

Semaphore (SM) registers allow a CPU to temporarily suspend program operation in order to synchronize operation with other CPUs. Each CPU can set or clear each SM register in its assigned cluster and can perform a test and set instruction on those SM registers. A test and set instruction can result in a CPU holding further execution of instructions until the appropriate SM register is cleared by another CPU assigned to the cluster. Each CPU in the cluster can also transmit all 32 SM registers to or from a scalar register.

Real-time Clock

The CRAY Y-MP C90 mainframe has a real-time clock (RTC) that increments synchronously with program execution and may be used to compute the running time for a program in clock periods (CPs). The RTC is a 64-bit counter that increments each CP.

CPU Computation Section

Each CPU is an identical, independent computation section consisting of operating registers, functional units, and an instruction control network (refer again to Figure 2-1). The operating registers and functional units store and process three types of data: address, scalar, and vector.

Address data controls internal operations and consists of information such as memory addresses, register designators and indexes. Address data is stored in the address (A) registers and intermediate address (B) registers and is processed in two dedicated functional units.

Scalar data is any discrete numerical quantity that can be processed in functional units either singly or in operand pairs to produce a single scalar result. Scalar data is stored in the scalar (S) registers and the intermediate scalar (T) registers and is processed in four dedicated functional units. Scalar floating-point data is processed in one of three floating-point functional units; these functional units are also used to process vector floating-point data.

Vector data refers to a set (or vector) of discrete numerical quantities that can be referenced by a single name. Vector data can be processed either singly or in operand pairs in special functional units to produce a vector result. Practically speaking, this means that a single instruction can result in the same operation being performed sequentially on a whole set of operands to produce a set of results. Vector data is stored in the vector (V) registers and is processed in five dedicated functional units. Vector floating-point data is processed in one of three floating-point functional units; these functional units are also used to process scalar floating-point data.

The 32-bit integer product is a vector instruction designed for index calculation. A full-indexing capability is possible throughout central memory in either scalar or vector modes. 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.

Data flow in a computation 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. Depending on the instruction sequence, data flows along either the scalar or vector path with two exceptions. In some cases, the scalar registers may provide one of the required operands for some vector operations performed in the vector functional units. Also, some scalar functional units return their results to an address register.

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.

Floating-point instructions allow addition, subtraction, multiplication, and reciprocal approximation operations. The reciprocal approximation instructions used in conjunction with other instructions enable floating-point division operations.

The instruction set includes logical operations for AND, inclusive OR, exclusive OR, exclusive NOR, and mask-controlled merge operations. Shift operations allow the manipulation of either 64-bit or 128-bit operands to produce 64-bit results. With the exception of 32-bit integer arithmetic performed in the A register functional units, most operations are used in vector or scalar instructions.

The following subsections describe the operating registers and their associated functional units.

Operating Registers

Each CPU has three primary and two intermediate sets of operating registers. The primary sets of operating registers are the address (A), scalar (S), and vector (V) registers. These registers are considered primary because functional units and central memory can access them directly.

For the A and S registers, an intermediate level of registers exists. The A registers are supported by the intermediate address (B) registers, and the S registers are supported by the intermediate scalar (T) registers. The B and T registers cannot access the functional units, and serve mainly as a memory buffer for the primary registers. To reduce the number of memory reference instructions for scalar and address operations, block transfers are possible between the B and T registers and central memory. The V registers do not have associated intermediate 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.

The data stored in B registers are protected with parity bits. When a word is written into a B register, a set of parity bits is generated and stored with the data bits. This set of parity bits is compared to another set that is generated when a word is read out of the B register. An error is indicated when the two sets do not match. Parity errors set the register parity error (RPE) flag in the exchange package if interrupt on register parity error (IRP) mode is set and enabled. They also report the location of the error to the status register.

Scalar Registers

Each CPU contains eight 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 of 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.

The data stored in T registers are protected with parity bits. When a word is written into a T register, a set of parity bits is generated and stored with the data bits. This set of parity bits is compared to another set that is generated when a word is read out of the T register. An error is indicated when the two sets do not match. Parity errors set the register parity error (RPE) flag in the exchange package if interrupt on register parity error (IRP) mode is set and enabled. They also report the location of the error to the status register.

Vector Registers

Each CPU contains eight V registers. Each V register contains 128_{10} elements; each element can store 64 bits of data. In vector operations, the 128 elements are processed in two groups, called pipes. One pipe processes the even-numbered elements while the other pipe simultaneously processes the odd-numbered elements. Each pipe is supported by an identical set of functional units.

The effective length of a V register for any operation is controlled by the program-selectable vector length (VL) register. The VL register is an 8-bit register that specifies the number of vector elements processed by the vector instructions. The contents range from 1_8 through 200_8.

The vector mask (VM) register allows for the logical selection of particular elements of a vector. The VM register is a 128-bit register; each bit corresponds to an element of a vector register. Bit 2^{127} corresponds to element 0 and bit 2^0 corresponds to element 127. The mask is used with vector merge and test instructions to allow operations to be performed on individual vector elements.

V register data is protected with parity bits. When a word is written into a V register, a set of parity bits is generated and stored with the data bits. This set of parity bits is compared to another set that is generated when the word is read out of the V register. An error is indicated when the two sets do not match. Parity errors set the register parity error (RPE) flag in the exchange package if interrupt on register parity error (IRP) mode is set and enabled. They also report the location of the error to the status register.

For more information on vector processing, refer to "Vector Processing" in this section.

Functional Units

Instructions other than simple transfers of data 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 all can operate simultaneously.

All functional units perform their specific operation in a fixed amount of time; delays are impossible once the operands are delivered to the unit. 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" in this section for more information on pipelining, segmentation, and functional unit independence.

There are four groups of functional units: address, scalar, vector, and floating-point. The address, scalar, and vector functional units operate with one of the primary register types (A, S, and V) to support address, scalar, and vector processing. The floating-point functional units support either scalar or vector operations and accept operands from or deliver results to S or V registers. For timing purposes, central memory can also act as a functional unit for vector operations.

The following subsections define the functions and the instructions executed by each functional unit. Refer to the following sections and subsections for additional information on functional units.

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 two address functional units are described below.

- 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 32 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. The exception is the population/parity/leading zero count functional unit, which delivers its result to an A register.

Four functional units are exclusively associated with scalar operations and are described below. Three floating-point functional units are used for both scalar and vector operations. Refer to "Floating-point Functional Units" in this section for more information on these units.

- The scalar add functional unit performs integer addition and subtraction; subtraction is performed by using two's complement arithmetic. Overflow is not detected.

- The scalar logical functional unit performs bit-by-bit manipulation of quantities obtained from S registers.

- The scalar shift functional unit shifts the entire contents of an S register (single shift) or shifts the contents of two concatenated S registers (double shift) into a single resultant S register. Single shifts are end-off with zero fill, while double shifts can be circular fill. Shift counts are obtained from an A register or from a field of the instruction.

- The scalar population/parity/leading zero count functional unit counts the number of 1 bits in an operand obtained from an S register and then, depending on the instruction issued, returns the count either as a population or population parity count to an A register. For the leading zero function, the unit counts the number of 0 bits preceding the first 1 bit in an operand obtained from an S register and returns the count to an A register.

Vector Functional Units

There are two parallel sets of vector functional units referred to as pipe 0 and pipe 1. Pipe 0 processes the even-numbered elements of a vector, while pipe 1 processes the odd-numbered elements. This duplication of functional units allows two pairs of elements to be processed at the same time and increases the efficiency of the vector processing operations.

Most vector functional units perform operations on operands obtained from one or two vector registers or from a vector register and an S register. The shift, population/parity, and leading zero functional units require only one operand. Results from a vector functional unit are delivered to a V register.

The functional units described in this section are used exclusively for vector operations. Three functional units are associated with both vector operations and scalar operations. Refer to "Floating-point Functional Units" in this section for more information on these 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. The subtraction operation uses two's complement arithmetic. Overflow is not detected.

- The vector shift functional unit shifts the entire contents of a vector register element (single shift) or the value formed from two consecutive elements of a V register (double shift). Shift counts are obtained from an A register and are end-off with zero fill.

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

- The second vector logical functional unit, when enabled, performs the same type of bit-by-bit manipulations as the full vector logical functional unit, but not for all instructions. The second vector logical functional unit cannot perform vector register merge, compressed index, and the logical operations associated with the vector mask instructions. A bit in the exchange package enables or disables the second vector logical functional unit.

- The vector population/parity/leading zero count functional unit performs population counts, parity checks, and leading zero counts for vector operations. These operations are identical to those performed in the scalar population/parity/leading zero count functional unit, except that the operands are the elements of a V register, and the results are returned to a V register.

Floating-point Functional Units

There are two parallel sets of floating-point functional units, with each set containing three functional units. These floating-point functional units perform floating-point arithmetic for both scalar and vector operations. The vector registers use both sets of functional units; one set processes the even-numbered elements, while the other set processes the odd-numbered elements. For an operation involving only scalar operands, only one set of floating-point functional units is used.

When executing most vector instructions, operands are obtained from pairs of V registers, or from an S register and a V register, and results are delivered to a vector register. When a floating-point functional unit is used for a vector operation, the general description of vector functional units applies. When executing a scalar instruction, operands are obtained solely from S registers, and results are delivered to an S register.

- The floating-point add functional unit performs addition and subtraction of operands in floating-point format. The result is normalized even when operands are unnormalized. The floating-point add functional unit detects overflow and underflow conditions; only overflow conditions are flagged.

- The floating-point multiply functional unit performs full- and half-precision multiplication of operands in floating-point format. The half-precision product is rounded; the full-precision product can be rounded or not rounded. This functional unit also generates a 32-bit integer product.

  Input operands must be normalized; the floating-point multiply functional unit delivers a normalized result only if both input operands are normalized. The floating-point multiply functional unit detects overflow and underflow conditions; only overflow conditions are flagged.

  The floating-point multiply functional unit recognizes both operands with zero exponents as a special case and performs an integer multiply operation. The result is considered an integer product, is not normalized, and is not considered 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 floating-point reciprocal approximation functional unit delivers a correct result only if the input operand is normalized. The high-order bit of the coefficient is not tested, but is assumed to be a 1. The floating-point reciprocal approximation functional unit detects overflow and underflow conditions; both conditions are flagged.

Functional Unit Operations

Functional units in a CPU perform logical operations, integer arithmetic, and floating-point arithmetic. Integer arithmetic and floating-point arithmetic are performed in two's complement. The following subsections explain the logical operations, the integer arithmetic, and the floating-point arithmetic used by the CRAY Y-MP C90 mainframe.

Logical Operations

Scalar and vector logical functional units perform bit-by-bit manipulation of 64-bit quantities. Instructions are provided for forming logical products, sums, exclusive ORs, equivalences, and merges.

A logical product is the AND function, which is shown in the following example:

```
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 in the following example:

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

A logical exclusive OR function is shown in the following example:

```
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 in the following example:

```
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. The bits of operand 1 are transmitted to the result when the mask bit is a 1. The bits of operand 2 are transmitted to the result when the mask bit is a 0. The following example shows a merge operation:

```
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 32 or 64 bits long, are represented in the registers as shown in Figure 2-2. The address add and address multiply functional units perform 32-bit arithmetic. The scalar add and vector add functional units perform 64-bit arithmetic.

Two scalar (64-bit) integer operands are multiplied 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.

The floating-point multiply functional unit recognizes a condition in which both operands have zero exponents as a special case. This case 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 (bits 2^{62} through 2^{48}) equal to 0 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's.

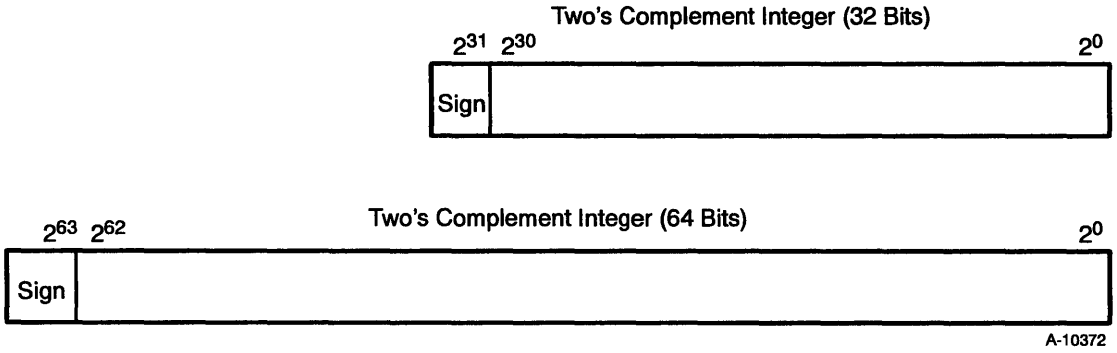


Figure 2-2. Integer Data Formats

When the floating-point multiply functional unit performs the operation, it returns the 48 high-order 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. If the 24 least significant bits of the operand coefficients were nonzero, the 48 low-order bits of the product could be nonzero and could generate a carry into the least significant of the 48 high-order bits returned, causing the result to be one larger than expected.

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} follows the rules for multiplying signs, and the result is a signed-magnitude integer. An exclusive OR function on bits 2^{63} of operands 1 and 2 is performed to derive the sign of the

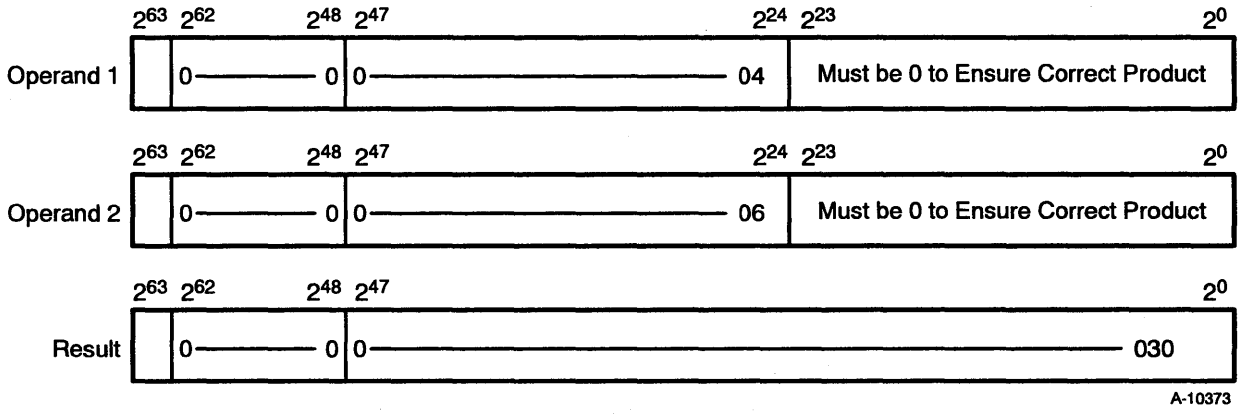


Figure 2-3. 24-bit Integer Multiply Performed in a Floating-point Multiply Functional Unit

result. The format of integers expected by both the hardware and software is two's complement, not signed-magnitude; therefore, negative products must be converted to two's complement form.

The second multiplication method is used when the operands are more than 24 bits long; multiplication is done by software, which forms multiple partial products and then shifts and adds the partial products.

A second integer multiplication operation performs a 32-bit multiplication operation on the S*j* operand and the V*k* operand and puts the result in the V*i* register. The operands must be shifted left before the operation begins. The S*j* operand must be shifted left 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 V*k* operand must be shifted left 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. Bits 2^{63} through 2^{48} are zero filled. The result of the multiply is right justified into positions 2^{31} through 2^{0}, and positions 2^{32} through 2^{63} are zero filled.

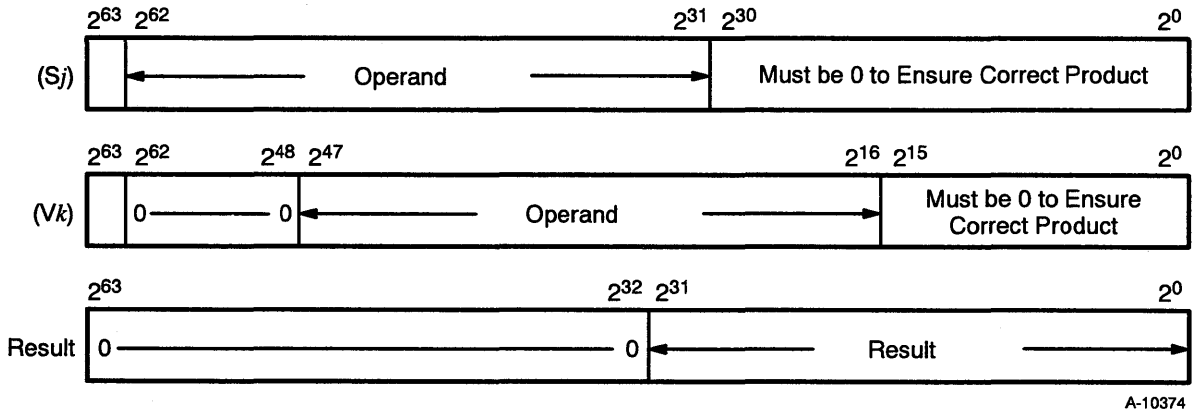


Figure 2-4. 32-bit Integer Multiply Performed in a Floating-point Multiply Functional Unit

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. For more information on integer division, refer to "Floating-point Division Algorithm" in this section.

Floating-point Arithmetic

The scalar and vector instructions use floating-point arithmetic. The following subsections explain floating-point arithmetic.

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 fields: coefficient sign, exponent, and coefficient.

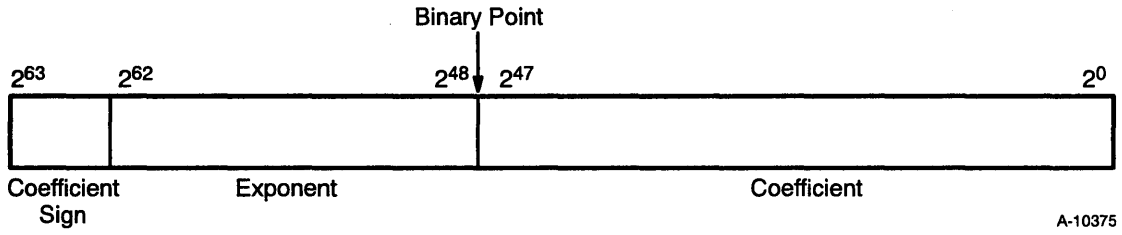


Figure 2-5. Floating-point Data Format

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. Figure 2-6 shows the biased and unbiased exponent ranges. Bit 2^{61} is the sign of the exponent; a 0 indicates a positive exponent, and a 1 indicates a negative exponent. Bit 2^{62} is the bias of the exponent. The floating-point format of the system allows the accurate expression of numbers to about 15 decimal digits in the approximate range of 10^{-2466} through 10^{+2466}.

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 signed-magnitude format, it is not complemented for negative values.

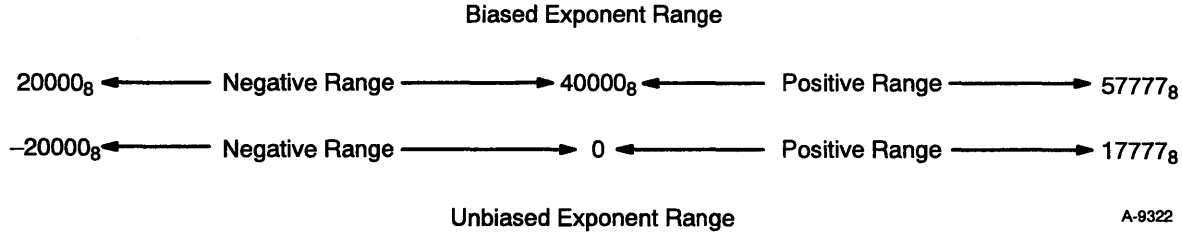


Figure 2-6. Biased and Unbiased Exponent Ranges

Figure 2-7 and the following steps show the relation between the biased exponent and the coefficient. The following steps show how to convert a floating-point number to its decimal equivalent.

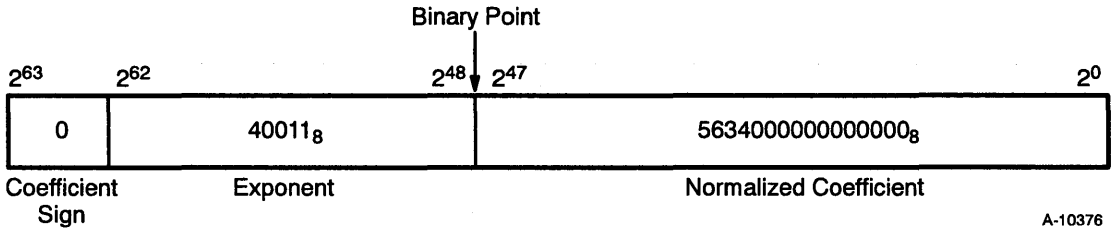


Figure 2-7. Internal Representation of a Floating-point Number

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

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

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 0's. A negative 0 is not generated by any floating-point functional unit, except in the case in which a negative 0 is one operand going into the floating-point multiply or floating-point add functional unit.

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 0's in its coefficient. The exception is a normalized floating-point 0, which is all 0's.

Anytime an integer is converted to a floating-point number, normalize the result before using it in a floating-point operation. Normalization is accomplished by adding the unnormalized floating-point operand to 0.

The reciprocal approximation functional unit must use normalized numbers to produce correct results. Using unnormalized numbers produces inaccurate results.

The floating-point multiply functional unit does not require the use of 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; unnormalized floating-point numbers may be routed through this functional unit to take advantage of this process.

Floating-point Range Errors

To ensure that the limits of the functional units are not exceeded, a range check is performed for overflow and underflow conditions on the exponent of each floating-point number coming into the functional unit. Bits 2^{61} and 2^{62} are checked; if both bits are equal to 1, the exponent is equal to or greater than 60000_8, and an overflow condition is detected.

When an overflow condition is detected, an interrupt occurs only if the interrupt-on-floating-point error (IFP) mode is set and enabled. In this case, the floating-point error (FPE) flag is set, causing an exchange sequence to occur. The IFP mode can be set or cleared by a user mode program.

When an overflow condition occurs, the value returned to the result register depends on the functional unit used. For the floating-point add and floating-point multiply functional units, the calculated coefficient, together with a forced exponent of 60000_8, is sent to the result register. For the reciprocal approximation functional unit, the returned result is the same except that bit 2^{47} of the coefficient is set to 0. Refer to Figure 2-8 and Figure 2-9.

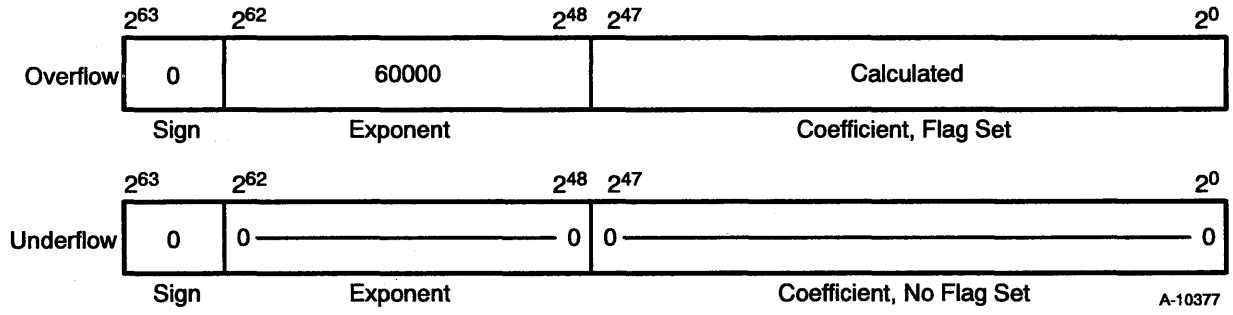


Figure 2-8. Floating-point Add and Floating-point Multiply Range Errors

To check for an underflow condition in the floating-point 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.

If an underflow condition is detected in the floating-point add or floating-point multiply functional unit, no fault is generated, and the word returned from the functional unit is all 0 bits. Refer to Figure 2-8. The floating-point multiply functional unit will not detect an underflow condition if both exponents equal 0; instead an integer multiply operation is performed.

Because the underflow condition of the result generated by the floating-point add functional unit is tested before the result is normalized, the normalized result can have a valid exponent as low as 17721_8. This occurs when the unnormalized result has an exponent of 20000_8 and a coefficient of 1. In this case, no underflow is detected, and the calculated result is sent to the result register.

An underflow condition is detected in the reciprocal approximation functional unit if either of the incoming operands has an exponent less than or equal to 20001_8. If this condition occurs, the FPE flag will set only if IFP mode is set and enabled. The calculated coefficient, with bit 2^{47} set to 0, together with a forced exponent of 60000_8, is sent to the result register. Refer to Figure 2-9.

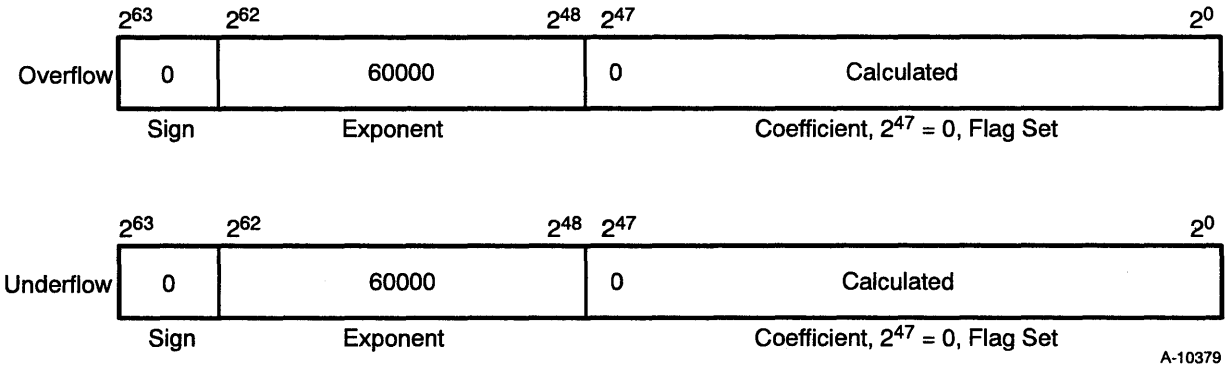


Figure 2-9. Floating-point Reciprocal Approximation Range Errors

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 results.

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

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

The last operation performed normalizes the results. To normalize the result, the coefficient is shifted left by the number of leading 0's (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 shifted right by 1 and the exponent increases by 1.

The normalization feature of the floating-point add functional unit is used to normalize any floating-point number. The number is simply paired with a zero operand and sent through the floating-point add functional unit.

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

Floating-point Multiplication Algorithm

The floating-point multiply functional unit receives two 48-bit floating-point coefficients from either an S or V register as input. Multiplication is commutative, that is, A x B = B x A. The signs of the two operands are combined by an exclusive OR function, the exponents are added together, and the two 48-bit coefficients are multiplied together. Multiplying the 48-bit coefficients produces a product of either

95 or 96 bits. A 96-bit product is normalized as it is generated, but a 95-bit product requires a left shift of 1 to generate the final normalized coefficient. If a shift occurs, 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. Some of the lower 48 bits are never generated. To adjust for this truncation, a constant is unconditionally added to the product. The average value of this truncation is 9.25×2^{-56}, which is determined by adding all carries produced by all possible combinations that could be truncated and dividing the sum by the number of possible combinations. Nine carries are inserted at bit position 2^{-56} to compensate for the truncated bits.

If the truncated bits are not compensated for, the resulting coefficient is 1 bit position smaller than expected. With compensation, the resulting coefficient ranges 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. Rounding is optional, but 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 propagate.

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 propagate upward, and the 29 most significant bits of the normalized result are transmitted back.

The result variations caused by this truncation and rounding are in the following ranges:

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

or

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

With a 96-bit product and rounding equal to one-half the least significant bit, the following result variation is expected:

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

Floating-point Division Algorithm

The CRAY Y-MP C90 mainframe does not have a single functional unit dedicated to the division operation. Rather, the floating-point multiply and reciprocal approximation functional units together carry out the algorithm. The following paragraphs explain the algorithm and how it is used in the functional units.

Finding the quotient of two floating-point numbers involves two steps. For example, to find the quotient A/B, first the B operand is sent through the reciprocal approximation functional unit to obtain its reciprocal, 1/B. Then, this result along with the A operand is sent to the floating-point multiply functional unit to obtain the product A x 1/B.

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. Refer to Figure 2-10.

To find the reciprocal, the equation $F(x) = 1/x - B = 0$ 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 $1/x - B = 0$. The 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). The initial approximation, 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 x-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:

$$x_{(i+1)} = 2x_i - x_i^2B = x_i (2 - x_iB)$$

In the equation, $x_{(i+1)}$ 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.

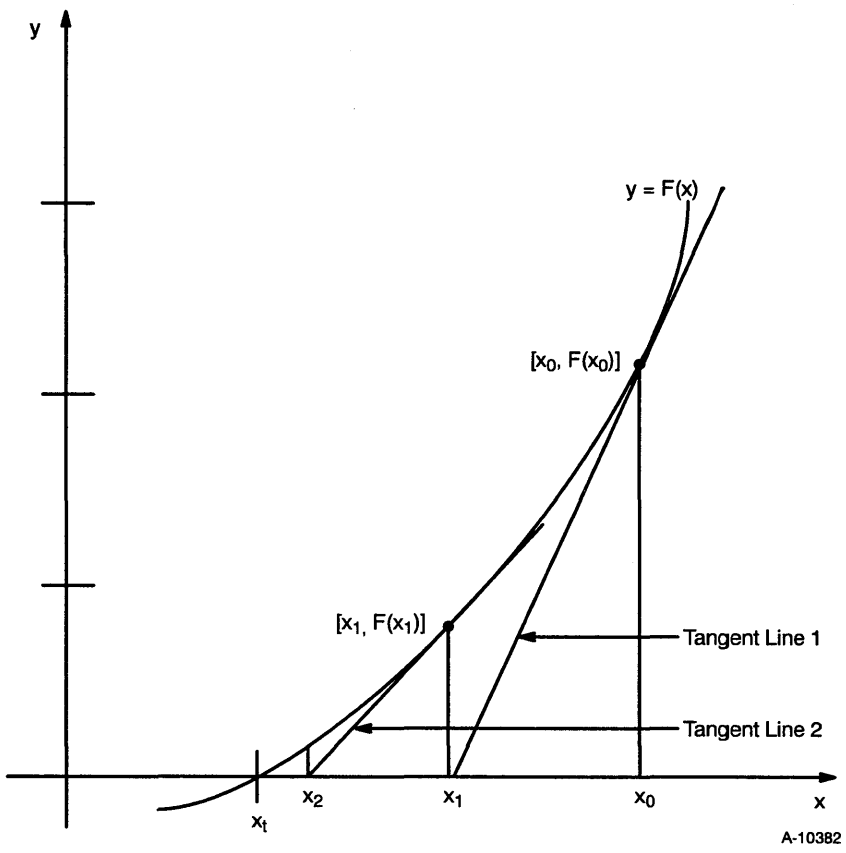


Figure 2-10. Newton's Method for Approximating Roots

The CRAY Y-MP C90 mainframe uses this approximation technique based on Newton's method. A hardware look-up table provides an initial guess, x_0, accurate to within 8 bits, 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 16 bits.
2	$x_2 = x_1(2 - x_1B)$	The second approximation is done in the reciprocal approximation functional unit and is accurate to 30 bits.
3	$x_3 = x_2(2 - x_2B)$	The third approximation is done in the floating-point multiply functional unit to calculate the correction term.

The reciprocal approximation functional unit calculates the first two iterations, while the floating-point multiply functional unit calculates the third iteration. The third 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.

The reciprocal iteration is designed for use once with each half-precision reciprocal generated. If the third iteration (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. A fourth iteration should not be done.

The following example shows how the floating-point multiply functional unit provides a full-precision result, computing the value of S1/S2.

Step	Operation	Unit
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 rounding

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 multiply to produce a half-precision quotient in only two operations, as shown in the following example.

Step	Operation	Unit
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 multiply results are returned as 0's with rounding applied to the low-order bit of the 29-bit result.

The following is another method of performing the division operation:

Step	Operation	Unit
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 numerator is multiplied by the half-precision reciprocal rather than before the multiplication.

The coefficient of the reciprocal produced by this alternative method can differ 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 as much as twice, while the other method may not round at all. The first rounding can occur while the correction is generated, and the second rounding can occur when the final quotient is produced. Therefore, the reciprocals should be compared using the same method each time they are generated. Cray Fortran CFT and CFT77 use a consistent method to ensure that the reciprocals of numbers are always the same.

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 Cray Research.

CPU Control Section

Each central processing unit (CPU) is assigned tasks and is controlled in the execution of those tasks through exchange sequences, fetch sequences, and issue sequences. These three sequences are closely related. For an initial deadstart program or a new program to run, an exchange sequence must occur. This sequence of steps sets several important parameters of the program in the CPU and may initialize some of the CPU's operating registers. A fetch sequence begins immediately after the exchange sequence and transfers a block of instructions from memory to an instruction buffer. The issue sequence then selects the instruction indicated by the program address (P) register, decodes it, determines whether the required registers or functional units are available, and if so, allows the instruction to be executed.

As the instruction executes, the P register increments, causing new instructions to be selected from an instruction buffer and to move through the issue sequence. When a desired instruction is not currently in an instruction buffer, another fetch sequence occurs, retrieving another block of instructions from memory. This overall process continues until either the program terminates or is interrupted, at which time another exchange sequence occurs and the entire process begins again.

The following subsections describe the exchange mechanism, the instruction fetch sequence, and the instruction issue sequence unique to each CPU. The programmable clock, the status register, and the performance monitor are also briefly described.

Exchange Mechanism

Each CPU uses an exchange mechanism for switching instruction execution from program to program. This exchange mechanism transfers blocks of program parameters (known as exchange packages) during a CPU operation, which is referred to as an exchange sequence.

The following subsections describe the contents of the exchange package and explain the exchange sequence in more detail.

Exchange Sequence

The exchange sequence moves the contents of an inactive exchange package from memory into the operating registers. Simultaneously, the exchange sequence retrieves data from the operating registers, uses it to construct the active exchange package, and then moves this exchange package back into memory. This swapping operation occurs in a fixed sequence when all computational activity associated with the active exchange package stops.

The exchange sequence involves 16 memory read references and 16 memory write references. A single 16-word block of memory data is used as the source of the inactive exchange package and the destination of the active exchange package. Word 0 of the active exchange package is swapped with word 0 of the inactive exchange package. The location of this block of data is specified by the contents of the XA register and is part of the active exchange package.

Exchange Package

An exchange package is a 16-word block of data stored in a reserved area of memory that contains the initial parameters for a particular computer program. In addition to initializing the program, these parameters also provide continuity if a program stops and restarts processing from one section of the program to the next.

The exchange package includes the contents of the address (A) and scalar (S) registers. The contents of the intermediate address (B), intermediate scalar (T), vector (V), vector mask (VM), shared B (SB), shared T (ST), and semaphore (SM) registers are not saved in the exchange package. Data in these registers must be stored and replaced as required by the program supervising the object program or by any program that needs this data.

Program Address Register Field

The program address (P) register contents are stored in the program address register field of the exchange package. There are 32 bits in the P register, the lower 2 bits of which are used to select a particular 16-bit parcel of a memory word. The P register is wide enough to address 1 gigaword of memory in C90 mode and 4 Mwords of memory in Y-MP mode.

The address stored in the P register field is the address of the first instruction that issues when the program that corresponds to this exchange package executes.

Instruction Base Address Register Field

The instruction base address (IBA) register holds the base address of the user's instruction area (the location in memory where a program's instruction area begins). The absolute memory address for an instruction fetch sequence is formed by adding the contents of the IBA register to the 30 high-order bits of the contents of the P register.

Instruction Limit Address Register Field

The instruction limit address (ILA) register holds the limit address of the program's memory image area, which is used to determine the highest absolute memory address that can be accessed during an instruction fetch sequence.

The absolute memory address used in an instruction fetch sequence must be an address between the IBA and ILA specified for the program being executed, or a program range error occurs. If the interrupt-on-program-range-error (IPR) mode is set in the exchange package, this error sets the program-range-error (PRE) interrupt flag. Regardless of the state of the IPR mode, a CPU interrupt will occur.

Data Base Address Register Field

The data base address (DBA) register holds the base address of the user's data area (the location in memory where a program's data area begins). Each time an instruction in the program makes a memory reference, the memory address generated by the instruction is added to the DBA to form the absolute memory address.

Data Limit Address Register Field

The data limit address (DLA) register holds the limit address of the user's data area, which is used to determine the highest absolute memory address the program can use for reading or writing data.

Each time an instruction makes a memory reference, the absolute memory address generated is compared to the DLA and the DBA. The absolute memory address must be between the DBA and the DLA, or an operand range error occurs. If the interrupt-on-operand-range-error (IOR) mode is set in the exchange package, this error sets the operand-range-error (ORE) interrupt flag, causing a CPU interrupt.

An instruction that attempts to read from a memory address outside the limits of the DBA and DLA still issues and finishes, but a zero value is transferred from memory. An instruction that attempts to write to a memory address ouside these limits issues, but no write operation occurs.

Interrupt Modes Field

There are 16 user-selectable interrupt modes, which allow the programmer to select the conditions under which the active program can be interrupted. These modes are usually selected in the exchange package, and with the exception of IPR, FEX, and FNX, they must be enabled by setting the EIM (enable interrupt modes) flag. The EIM flag sets automatically on an exchange to non-monitor mode and clears on an exchange back to monitor mode. While in monitor mode, the EIM flag can be set or cleared by instructions 001302 or 001303, respectively.

The interrupt modes are explained briefly in Table 2-1.

Table 2-1. CRAY Y-MP C90 Interrupt Modes

Mode	Description
IRP	Enables an interrupt if a register parity error is detected while reading data from a register.
IUM	Enables an interrupt if an uncorrectable memory error is detected while reading data from memory.
IFP	Enables an interrupt if a floating-point error occurs.
IOR	Enables an interrupt if an operand range error occurs.
IPR	Enables the PRE interrupt flag to set if a program range error occurs. A program range error always causes an exchange, regardless of the state of IPR. This mode is not affected by the EIM flag.
FEX	Enables the EEX interrupt flag to set if an error exit instruction (000000) issues. Issuing an error exit instruction always causes an exchange, regardless of the state of FEX. This mode is not affected by the EIM flag.
IBP	Enables an interrupt if a breakpoint occurs.
ICM	Enables an interrupt if a correctable memory error is detected while reading data from memory.
IMC	Enables an interrupt if requested by the maintenance control unit (MCU). The MCU for the CRAY Y-MP C90 is the MWS-E.
IRT	Enables an interrupt if requested by the real-time clock.
IIP	Enables an interprocessor interrupt if requested by another CPU.
IIO	Enables an I/O interrupt if SIE is set and this CPU is the lowest-numbered CPU with IIO=1 and EIM=1.
IPC	Enables an interrupt if requested by the programmable clock.
IDL	Enables an interrupt if a deadlock occurs while the program is not in monitor mode. IDL has no effect in monitor mode.

Table 2-1. CRAY Y-MP C90 Interrupt Modes (continued)

Mode	Description
IMI	Enables an interrupt if a monitor mode instruction (001*ijk*; $j \neq 0$) issues while the program is not in monitor mode. IMI has no effect in monitor mode.
FNX	Enables the NEX interrupt flag to set if a normal exit instruction (004000) issues. Issuing a normal exit instruction always causes an exchange, regardless of the state of FNX. This mode is not affected by the EIM flag.

Interrupt Flags Field

There are 16 interrupt flags, with one flag corresponding to each of the 16 user-selectable interrupt modes. If a particular interrupt mode (except IPR, FEX, or FNX) is set and enabled and the specified error occurs, the corresponding interrupt flag is set, forcing an exchange. If the error occurs while the appropriate interrupt mode is set but not enabled, the interrupt is held. This condition can occur only while the program is in monitor mode. Enabling the interrupt modes, either by exchanging to user mode or by issuing instruction 001302, enables the held interrupt to be processed, at which time it sets the corresponding interrupt flag and forces an exchange.

All interrupts or held interrupts, except PCI and ICP, are cleared on any exchange. PCI and ICP interrupts are held until they are cleared by instruction 001405 or 001402, respectively.

Two interrupt flags, deadlock (DL) and monitor instruction interrupt (MII), will set only if the corresponding interrupt modes are set and if the program is in non-monitor mode when the error occurs.

The I/O interrupt (IOI) flag sets only if the system I/O interrupts enabled (SIE) flag is set and if the CPU to be interrupted is the lowest-numbered CPU with IIO interrupt mode set and enabled. The SIE flag can be set by any CPU issuing instruction 001600. After any CPU is interrupted by an I/O interrupt, this flag is cleared, disabling all I/O interrupts. The interrupted CPU resets the SIE flag by issuing instruction 001600 after it has serviced the I/O interrupt.

Three errors always cause an exchange, regardless of the status of the EIM flag: a program range error, issuing instruction 000000, or issuing instruction 004000. The interrupt modes specifying these errors (IPR, FEX, and FNX) are used solely to enable setting the corresponding interrupt flags (PRE, EEX, and NEX respectively) should the appropriate error occur. Setting an interrupt flag in these cases makes it easier to determine the source of the error.

The errors that set interrupt flags are explained briefly in Table 2-2.

Table 2-2. CRAY Y-MP C90 Interrupt Flags

Flag	Description
RPE	The register parity error flag sets if the IRP interrupt mode bit is set and enabled and a parity error occurs during a read operation from a B, T, V, SB, or ST register or from an instruction buffer.
MEU	The memory error (uncorrectable) flag sets if the IUM interrupt mode bit is set and enabled and an uncorrectable memory error occurs while reading data from memory.
FPE	The floating-point error flag sets if the IFP interrupt mode is set and enabled and a floating-point range error occurs in any of the floating-point functional units.
ORE	The operand range error flag sets if the IOR interrupt mode bit is set and enabled and a data reference is made outside the address boundaries specified in the DBA and DLA registers.
PRE	The program range error flag sets if the IPR interrupt mode bit is set and enabled and an instruction fetch is made outside the address boundaries specified in the IBA and ILA registers. A program range error always causes an exchange, regardless of the state of IPR.
EEX	The error exit flag sets if the FEX interrupt mode bit is set and enabled and an error exit instruction (000000) issues. Issuing an error exit instruction always causes an exchange, regardless of the state of FEX.
BPI	The breakpoint interrupt flag sets if the IBP interrupt mode bit is set and enabled and a write reference is made to an address within the breakpoint range.
MEC	The memory error (correctable) flag sets if the ICM interrupt mode bit is set and enabled and a correctable memory error occurs while reading data from memory.
MCU	The MCU interrupt flag sets if the IMC interrupt mode bit is set and enabled and the MCU interrupt signal becomes active on I/O channel 40.
RTI	The real-time interrupt flag sets if the IRT interrupt mode bit is set and enabled and a real-time interrupt request is received.
ICP	The interprocessor interrupt flag sets if the IIP interrupt mode bit is set and enabled and another CPU requests an interrupt of this CPU by issuing instruction 0014j1.
IOI	The I/O interrupt flag sets if the SIE bit is set and this CPU is the lowest-numbered CPU with IIO interrupt mode set and enabled when a LOSP or VHISP channel completes a transfer.
PCI	The programmable clock interrupt flag sets if the IPC interrupt mode bit is set and enabled and the counter in the programmable clock equals 0.
DL	The deadlock interrupt flag sets if the IDL interrupt mode bit is set, the program is not in monitor mode, and a deadlock condition occurs because all CPUs in a cluster are holding issue on a test and set instruction.
MII	The monitor instruction interrupt flag sets if the IMI interrupt mode bit is set and a monitor mode instruction (001ijk; $j \neq 0$) issues while the program is not in monitor mode.
NEX	The normal exit flag sets if the FNX interrupt mode bit is set and enabled and a normal exit instruction (004000) issues. Issuing a normal exit instruction always causes an exchange, regardless of the state of FNX.

Status Field

The status field contains 4 bits used to indicate the state of the CPU at the time an exchange occurs. These status bits are set during program execution and are therefore not user selectable. Table 2-3 briefly describes each of the status bits used.

Table 2-3. CRAY Y-MP C90 Status Field Bit Assignments

Status	Description
VNU	The vectors not used bit sets if no vector instructions (077*ijk* or 140*ijk* through 177*ijk*) were issued during the execution interval.
FPS	The floating-point status bit sets if a floating-point error occurred during the execution interval.
WS	The waiting on semaphore bit sets if a test and set instruction (0034*jk*) is holding issue in the CIP register.
PS	The program state bit is set by the operating system to denote whether a CPU concurrently processing a program with another CPU is the master or slave in a multitasking situation.

Modes Field

There are four user-selectable modes that allow the programmer to select several modes of operation for the program. These modes are described briefly in Table 2-4.

Table 2-4. CRAY Y-MP C90 Operating Modes

Mode	Description
C90	If C90 mode is set, the program can use the full CRAY Y-MP C90 instruction set; otherwise, only CRAY Y-MP instructions can be executed.
ESL	If enable second vector logical mode is set, the second vector logical functional unit is enabled, and if it is not busy, it has first priority to execute instructions 140*ijk* through 145*ijk*.
BDM	If bidirectional memory mode is set, block read and write operations can operate concurrently.
MM	If monitor mode is set, the program can execute those instructions that are privileged to monitor mode.

Processor Number Field

The contents of the 4-bit processor number field indicate which CPU performed the exchange sequence. This value is not initially stored in the exchange package before the program starts; it is a constant value inserted into the exchange package after the program has run and been exchanged.

Cluster Number Field

The 5-bit cluster number (CLN) field contains the number to be loaded into the CLN register. This number selects one of 17_{10} available clusters of shared registers that the CPU can access. If the contents of the CLN register are 0, the CPU does not have access to any shared registers. The contents of the CLN registers in all CPUs are also used to determine a deadlock interrupt condition.

Exchange Address Register Field

The 8-bit exchange address (XA) register field specifies the address of the first word of a 16-word exchange package loaded by an exchange sequence. The XA register contains only the 8 high-order bits of a 12-bit absolute memory address. The low-order bits of the address are always 0, because an exchange package must begin on a 16-word boundary. The 12-bit limit on the absolute memory address means that all exchange packages are located in the lower 4,096 (10000_8) words of memory.

Vector Length Register Field

The 8-bit vector length (VL) register field specifies the length of all vector operations performed by vector instructions and the effective number of elements held in the V registers. The value in the VL register can be changed during program execution by using the 00200*k* instruction.

A Register Fields

The current contents of all A registers are stored in bits 2^{32} through 2^{63} of words 0 through 7 during an exchange sequence.

S Register Fields

The current contents of all S registers are stored in bits 2^{0} through 2^{63} of words 8 through 15 during an exchange sequence.

Instruction Fetch Sequence

An instruction fetch sequence retrieves program code from memory and places it in an instruction buffer. The program code is held in the instruction buffer before being delivered to the instruction issue registers.

Instruction Issue

An instruction issue sequence is the series of steps performed to move an instruction from an instruction buffer through the issue registers and into execution.

Programmable Clock

Each CPU has a programmable clock that generates periodic interrupts at specific preset intervals. Available intervals range between 9 and $2^{32}-1$ CPs. Intervals shorter than 100 μs are not practical because of the monitor overhead involved in processing the interrupt. These instructions are privileged to monitor mode.

Status Registers

Each CPU contains eight status registers. Memory error and register parity error information is reported to these registers, as well as information on the status of several bits in the active exchange package.

Performance Monitor

The performance monitor tracks groups of hardware-related events. These results can be used to indicate the relative performance of a program. The performance monitor contains thirty-two 48-bit performance counters.

Performance events are monitored only when operating in non-monitor mode. Entering monitor mode disables the performance counters.

Two types of instructions are used with the performance monitor: user instructions and maintenance instructions. The user instructions allow the user to select and read the performance monitor. The maintenance instructions test the logic of the performance monitor.

Parallel Processing Features

The CRAY Y-MP C90 mainframe has several special features that enhance the parallel processing capabilities inherent in the system. Parallel processing can mean different things in different environments; the following subsections discuss two types of parallel processing used:

- Parallel processing within a single CPU
- Parallel processing between two or more CPUs

Parallel processing features within a single CPU include instruction pipelining and segmentation, functional unit independence, vector processing (described earlier in this section), multitasking, and Autotasking. The first two features are inherent hardware features of the CRAY Y-MP C90 mainframe; a programmer has little control over these features. The vector processing feature can be manipulated by the programmer to provide optimum throughput. Refer to the "Vector Processing" subsection later in this subsection for more information on vector processing.

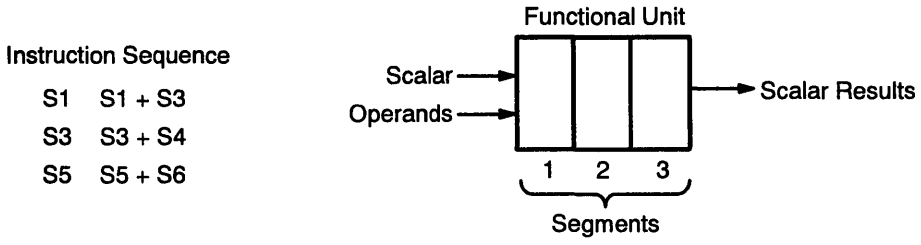
Parallel processing between two or more CPUs is called multiprocessing: the capability for several programs to run concurrently on multiple CPUs of a single mainframe. Included in this category are multitasking and the Autotasking feature of the CF77 Fortran compiling system. Multitasking is the capability to run two or more parts (or tasks) of a single program in parallel on different CPUs within a mainframe. Autotasking is automatic multiprocessing; it enables user programs to be automatically partitioned over multiple CPUs.

Pipelining and Segmentation

Pipelining means operation or instruction begins before a previous operation or instruction finishes. Pipelining is accomplished using fully segmented hardware. Segmentation means an operation is divided into a discrete number of sequential steps, or segments. Fully segmented hardware is designed to perform one segment of an operation during a single CP. At the beginning of the second CP, the partial results are sent to the second hardware segment in order to process the second step of the operation. During this second CP, the first hardware segment begins to perform the first step of the next operation.

In the CRAY Y-MP C90 mainframe, all hardware is fully segmented. Therefore, pipelining occurs during all hardware operations such as 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 the pipelining of three sets of scalar instructions through a segmented functional unit.



CP	Functional Unit Segment 1	2	3
1	S1 Partial Result		
2	S3 Partial Result	S1 Partial Result	
3	S5 Partial Result	S3 Partial Result	S1 Partial Result
4		S5 Partial Result	S3 Partial Result

A-10384

Figure 2-11. Scalar Segmentation and Pipelining Example

In the first CP, the first set of operands 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 the second pair of operands enters the first segment. This process continues each CP until the three operand pairs are completely processed. After 3 CPs, the first result leaves the functional unit and enters scalar register S1; the S3 and S5 results will be available in successive CPs.

The CRAY Y-MP C90 mainframe contains two sets of vector functional units: one for processing even-numbered elements and one for processing odd-numbered elements. This enables two pairs of elements to be processed in a single CP and almost doubles the vector processing rate. Figure 2-12 shows how a set of vector elements is pipelined through a dual vector functional unit. In the first CP, element 0 of register V1 and element 0 of register V2 enter the first segment of the pipe 0 functional unit, while element 1 of each register enters the pipe 1 functional unit. During the next CP, the partial results move to the second segments of each functional unit, while element 2 of both vector registers enters the first segment of the pipe 0 functional unit, and element 3 of both vector registers enters the first segment of the pipe 1 functional unit. This process continues each CP until all elements are completely processed.

In this example, the functional units are divided into five segments; the dual functional units process up to ten different pairs of elements simultaneously. After 5 CPs, the first results leave the functional units and enter vector register V3; subsequent results are available at the rate of two results per CP.

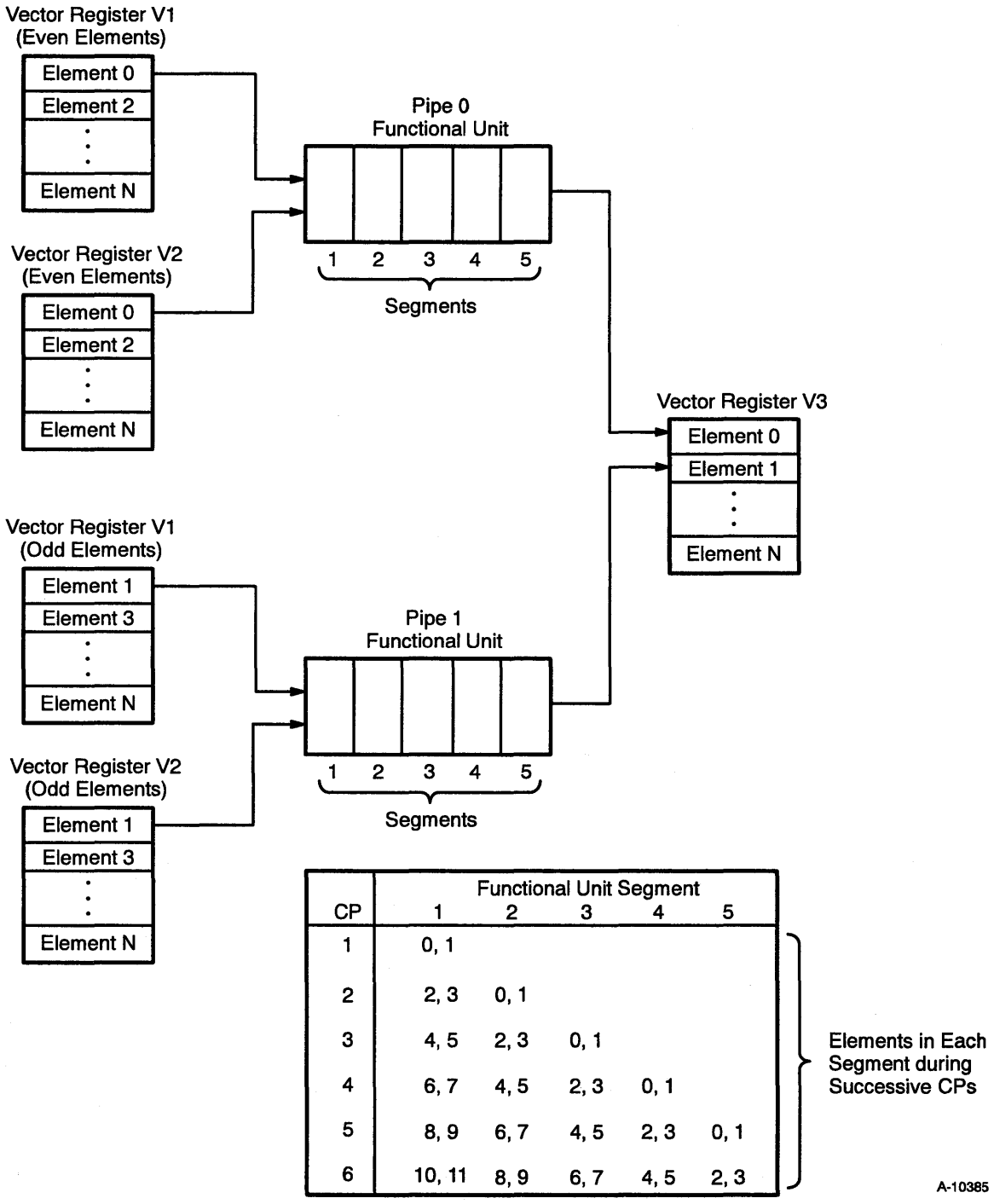


Figure 2-12. Vector Segmentation and Pipelining Example

Functional Unit Independence

The specialized functional units in the CRAY Y-MP C90 mainframe handle the arithmetic, logical, and shift operations. Most functional units are fully independent; any number of functional units can process instructions concurrently. Functional unit independence allows different operations such as multiplications and additions to proceed in parallel.

For example, the operation represented by the equation $A = (B + C) \times D \times E$ could be accomplished as follows. If operands B, C, D, and E are 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 multiplication operations proceed concurrently. Because the addition takes less time to run than the multiplication, both operations finish at the same time. The addition operation does not require additional processing time because it occurs during the same time interval as the multiplication operation. The results of these two operations are then multiplied to obtain the final result.

Vector Processing

Vector processing increases processing speed and efficiency by allowing an operation to be performed sequentially on a set (or vector) of operands through the execution of a single instruction.

A vector is an ordered set of elements; each element 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 in that it operates on many elements rather than on one.

In vector processing, two successive pairs of elements are processed each CP. The dual vector pipes and the dual sets of vector functional units allow a pair of even-numbered elements and a pair of odd-numbered elements to be processed during the same CP. As each pair of operations is completed, the results are delivered to successive even- or odd-numbered elements of the result register. The vector operation continues until the number of elements processed by the instruction equals the count specified by the vector length (VL) register.

Parallel vector operations allow the generation of more than two results per CP. Parallel vector operations occur automatically in the following situations:

- When successive vector instructions use different functional units and different V registers.
- When successive vector instructions use the result stream from one vector register as the operand of another operation using a different functional unit. This process is known as chaining and is explained later in this subsection.

Advantages of Vector Processing

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

Vectorization typically speeds up a code segment by an approximate factor of ten. If a segment of code that previously accounted for 50% of a program's running time is vectorized, the overall running time is 55% of the original running time (50% for the unvectorized portion plus 0.1 x 50% for the vectorized portion). Vectorizing 90% of a program causes running time to drop to 19% of the original execution time.

V Register Functions

The V registers are used solely for vector processing. This is unlike the A and S registers, which are used for many secondary functions. Vector processing allows a single instruction to perform a specified operation sequentially on a set (vector) of operands, to produce a vector of results. Examples of these sets or vectors may be rows or columns of a matrix or elements of a table.

The contents of a V register are transferred to or from central memory by means of a block transfer. A vector block transfer is accomplished by specifying a first word address in central memory, an increment or decrement value for the central memory address, and a vector length. The transfer begins with the first element of the V register and proceeds at a maximum rate of two words per clock period (CP); this rate can be affected by central memory conflicts. A central memory conflict interrupts the vector data stream and can occur in chained operations

(although they do not inhibit chaining). Any interruption in the vector data stream adds proportionally to the total execution time of vector operations.

Single-word data transfers can also be made between an S register and an element of a V register.

Vector Instructions

Vector instructions reference V registers by specifying the register number in the *i*, *j*, or *k* field of the instruction. Refer to the "Instruction Formats" subsection later in this section for information on instruction fields. Operations on vector registers always start with element 0. Individual elements of a V register are designated by octal numbers ranging from 00 through 177. These numbers appear as subscripts to vector register references. For example, $V6_{27}$ refers to element 27 of V register 6.

Vector instructions reserve V registers as either operands or results. If the register is reserved as an operand, it cannot be used as an operand or result until the operand reservation clears. A vector register can be used as both an operand and result register for the same vector instruction. If a register is reserved as a result, it can be used as an operand through a process called chaining. Refer to the subsection "Vector Chaining" in this section for more information on chaining.

No reservation is placed on the VL register during vector processing. If a vector instruction uses an S register as an operand, no reservation is placed on the S register. Conflicts can occur between vector and scalar operations involving floating-point operations and memory access. With the exception of these operations, the floating-point functional units are always available for scalar operations. The S and VL registers can be modified after the vector instruction issues without affecting the vector operation. The A0 and A*k* registers in a vector memory reference can also be modified after the instruction issues.

Because most transfers to or from registers are done in blocks of data, instructions that transfer data between V registers and central memory reserve a port, and functional unit instructions reserve the appropriate functional unit.

Vector Chaining

The CRAY Y-MP C90 mainframe 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 amount of concurrency in a chained operation depends on the relation between the issue time of the chaining instruction and the arrival time of the result data stream. For full chaining to occur, the chaining instruction must issue 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 of the result vector data stream.

Figure 2-13 shows how the results of four instructions are chained together. The instruction chaining sequence performs the following operations:

1. Reads a vector of integers from central memory to register V0.
2. Adds the contents of register V0 to the contents of register V1 and sends the results to register V2.
3. Shifts the results obtained in Step 2 and sends the results to register V3.
4. Forms the logical product of the shifted sum obtained in Step 3 with the contents of register V4 and sends the results to register V5.

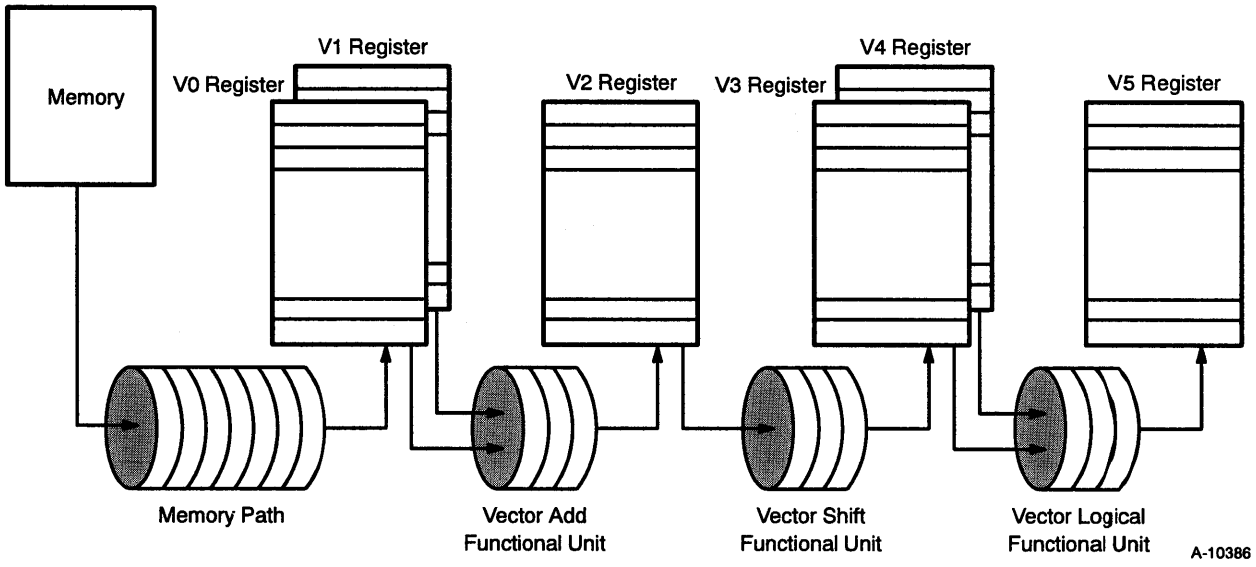


Figure 2-13. Vector Chaining Example

As soon as the first two elements from central memory arrive at register V0, they are added to the first two elements of vector register V1. Subsequent pairs of elements are pipelined through the segmented functional unit, so that a continuous stream of results is sent to the destination register, which is register V2. As soon as the first two elements arrive at register V2, they are used as operands for the shift operation. The results are sent to register V3, which immediately becomes the source of one of the operands necessary for the logical operation between registers V3 and V4. The results of the logical operation are then sent to register V5.

Multiprocessing and Multitasking

Users of CRAY Y-MP C90 mainframes can take advantage of parallel processing features known as multiprocessing and multitasking; this category also includes microtasking.

Parallel processing between two or more CPUs is called multiprocessing: the capability for several programs to be run concurrently on multiple CPUs of a single mainframe. Up to *n* programs can run simultaneously on a machine with *n* CPUs.

Multitasking is a more recent and complex enhancement than vectorization. Multitasking is the capability to run two or more parts, or tasks, of a single program in parallel on different CPUs within a mainframe. To take advantage of this feature, a program must be logically or functionally divided to allow two or more tasks to run simultaneously (that is, in parallel). For example, a weather modeling application in which the northern hemisphere calculation is one section of code and the southern hemisphere is another section of code. Distinct code segments are not needed; the same code could run on multiple processors simultaneously, with each processor handling separate data. Theoretically, the gain from multitasking can be calculated in the following manner. A program running on a dedicated system in wall clock time (*t*) could run in a time as short as *t/n* if multitasked, or modified to use *n* or more parallel tasks on a machine with *n* CPUs.

Actually, a speed-up factor of *n* is not quite attainable because of the additional processing operations (overhead) needed to implement multitasking. In some instances, multitasking can actually increase a program's execution time if the multitasking overhead decreases performance more than parallel processing time improves it. This is a situation that must be investigated before investing too much time and effort.

The following list includes some factors that limit the maximum improvement potential of a program:

- When not all parts of a program can be divided into parallel tasks.
- When those parts that can be multitasked may have dependencies on one another that result in one or more tasks having to wait for other tasks to finish.
- When the multitasking feature incurs additional processing time that is added to the program.

The CFT compiler on the CRAY Y-MP C90 mainframe automatically uses the vector hardware to perform operations on inner DO loops that have no data dependencies. Once such optimizing is complete, a single processor can work no faster, but more than one processor could operate on separate parts of the data simultaneously to achieve results faster. Microtasking permits multiple processors to work on a Fortran program at the DO-loop level. The name microtasking was chosen because multiprocessing is efficient even at a DO-loop level where the task size, or granularity, may be small.

Microtasking also works well when the number of processors available is unknown or may vary during program execution. This means that microtasked jobs do not require a dedicated system, although they perform best in a dedicated environment with no competing jobs.

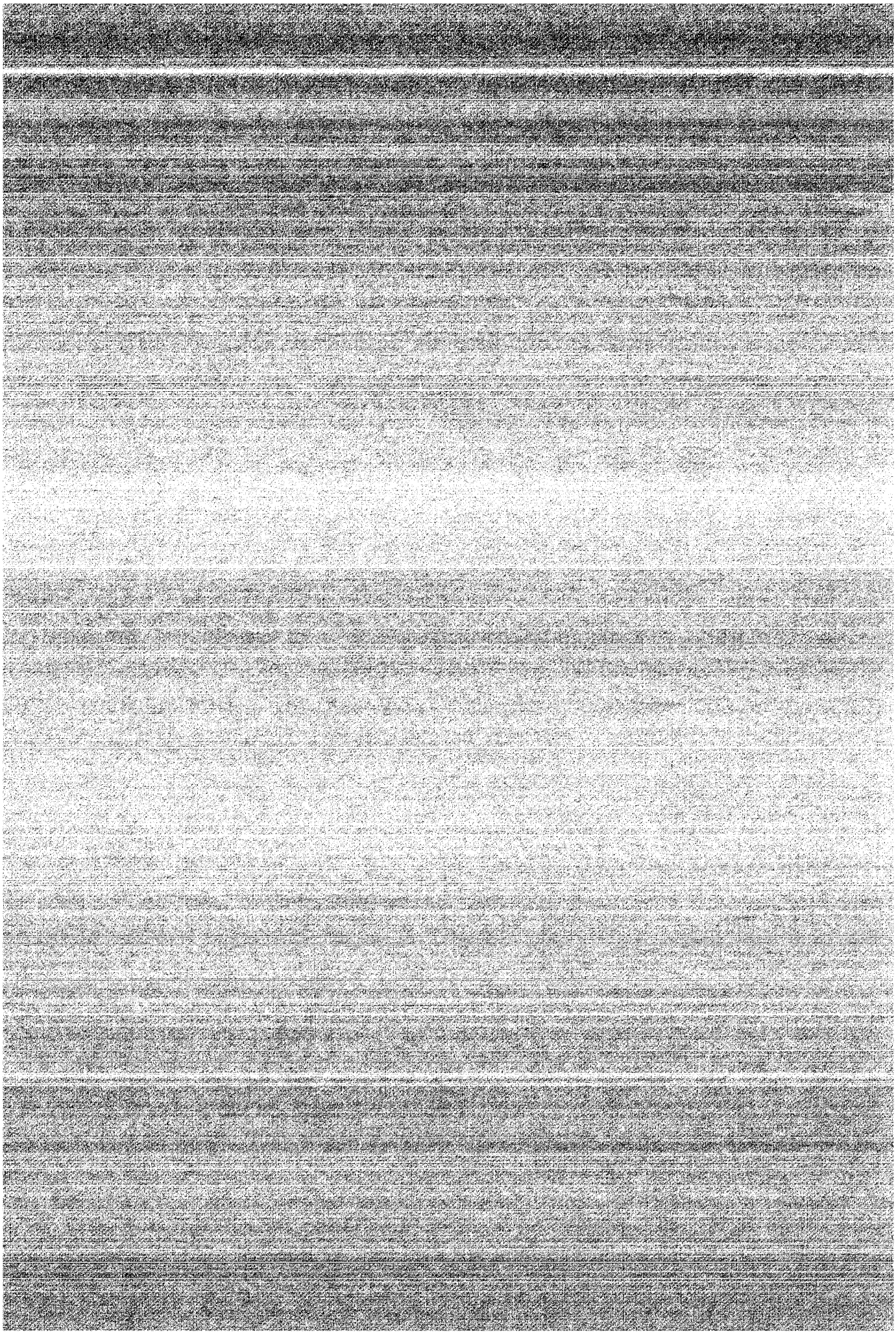
As stated before, advanced programming skills and tools are needed to use multiprocessing, multitasking, and microtasking concepts efficiently.

Autotasking

CRAY Y-MP C90 mainframe analysts and programmers can use Autotasking (automatic multitasking) to automatically detect whether portions of their programs can be run in parallel. Autotasking is an extension of multiprocessing and microtasking and is designed to make parallel processing easier to use. Autotasking alters a Fortran program to allow it to run simultaneously on multiple CPUs.

Autotasking is available on CRAY Y-MP computer systems beginning with UNICOS release 4.0 and CF77 release 3.0. Refer to *CF77 Compiling System, Volume 4: Parallel Processing Guide*, CRI publication number SG-3074, for more detailed information on Autotasking.

CPU INSTRUCTIONS



CPU Instructions

The following subsections explain the instruction formats, instruction differences between the Y-MP mode and C90 mode, special register values, special Cray Assembly Language (CAL) syntax forms, and monitor mode instructions used by the CRAY Y-MP C90 computer system. A CPU instruction summary is also included.

Notational Conventions

The following conventions are used throughout this section:

- All numbers are decimal numbers unless otherwise indicated.
- Letters X or x or *x* represent an unused value.
- Register bits are numbered from right to left as powers of 2.
- The letter n represents a specified value.
- The notation (value) specifies the contents of a register or memory location as designated by value.
- Variable parameters are in *italic* type.
- The vector mask bits are contained in the VM and VM1 registers. The bits of the VM register correspond to vector elements 0 through 63, and the bits of the VM1 register correspond to vector elements 64 through 127, as shown in Figure 2-14.

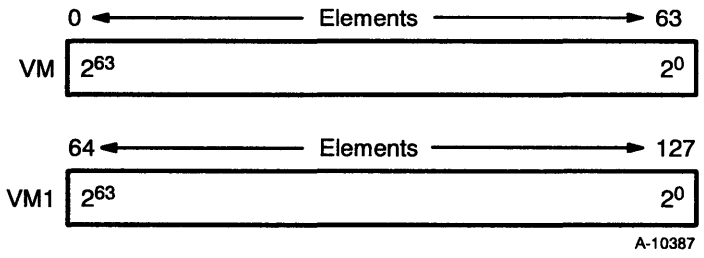


Figure 2-14. Vector Mask Bits

Instructions can be 1 parcel (16 bits), 2 parcels (32 bits), or 3 parcels (48 bits) 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 by branch instructions. A 2- or 3-parcel instruction can begin in any parcel of a word and can span a word boundary. For example, a 2-parcel instruction beginning in parcel 3 of word 1 ends in parcel 0 of

word 2. Parcels 0, 1, and 2 of word 1 do not need to be filled with all zeros or ones (padded). Figure 2-15 shows the general instruction format.

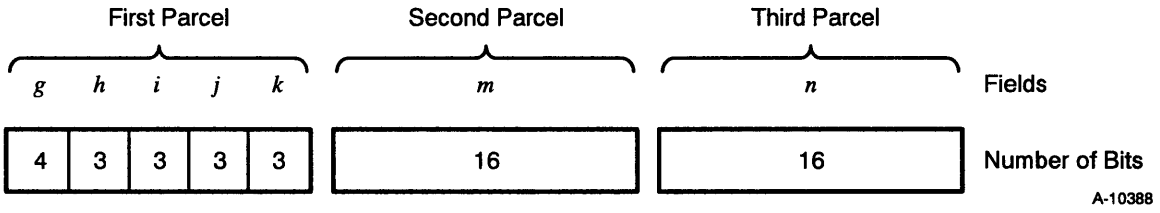


Figure 2-15. General Instruction Format

The first parcel is divided into five fields, and the second and third parcels each contain a single field. The four variations of this general format are listed below.

- 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 *i*, *j*, *k*, and *m* fields (Y-MP mode only)
- 3-parcel instruction format with combined *m* and *n* fields

Each format uses the fields differently and is described in detail in the following subsections.

Instruction Formats

The following subsections explain the instruction formats, as well as the instruction differences between Y-MP mode and C90 mode.

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 2-16). 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.

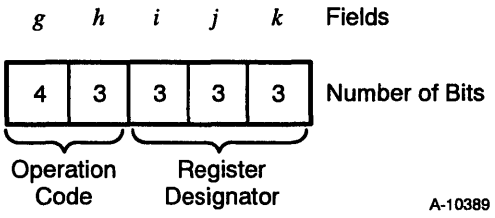


Figure 2-16. 1-parcel Instruction Format with Discrete *j* and *k* Fields

The following types of instructions use this format:

- Arithmetic
- Logical
- Vector shift
- Scalar double shift
- Floating-point constant

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 2-17). The *g* and *h* fields contain the operation code, and the *i* field usually designates a result register. The combined *j* and *k* fields contain a constant or an intermediate address (B) or intermediate scalar (T) register designator. The 005*ijk* branch instruction and the following types of instructions use the 1-parcel instruction format with combined *j* and *k* fields:

- 6-bit constant
- B or T register block memory transfer
- B or T register data transfer with address (A) or scalar (S) register
- Scalar single shift
- Scalar mask

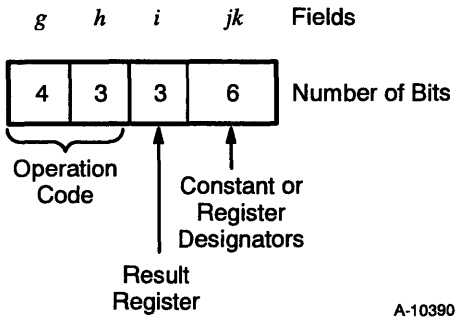


Figure 2-17. 1-parcel Instruction Format with Combined *j* and *k* Fields

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

The 2-parcel instruction 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 2-18). A 7-bit operation code (*gh*) is followed by an *ijkm* field. The high-order bit of the *i* field (*i*2) is equal to 0.

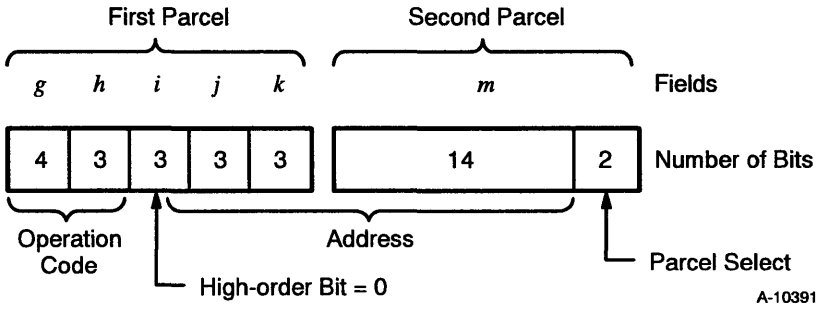


Figure 2-18. 2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields

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

There are three distinct 3-parcel instruction formats using the 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 instructions using this format transfer 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 for a C90 mode branch instruction uses the combined *m* and *n* fields to hold the memory branch address. The C90 mode is explained in the next subsection. The 7-bit *g* and *h* fields (and, in one case, bit 2^2 of the *i* field) contain an operation code.

The format for A or S register memory references uses the combined *m* and *n* fields to hold the memory reference address. 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 2-19 shows the three applications for the 3-parcel instruction format with combined *m* and *n* fields. Remember that the *m* and *n* fields are reversed when a 3-parcel instruction is assembled.

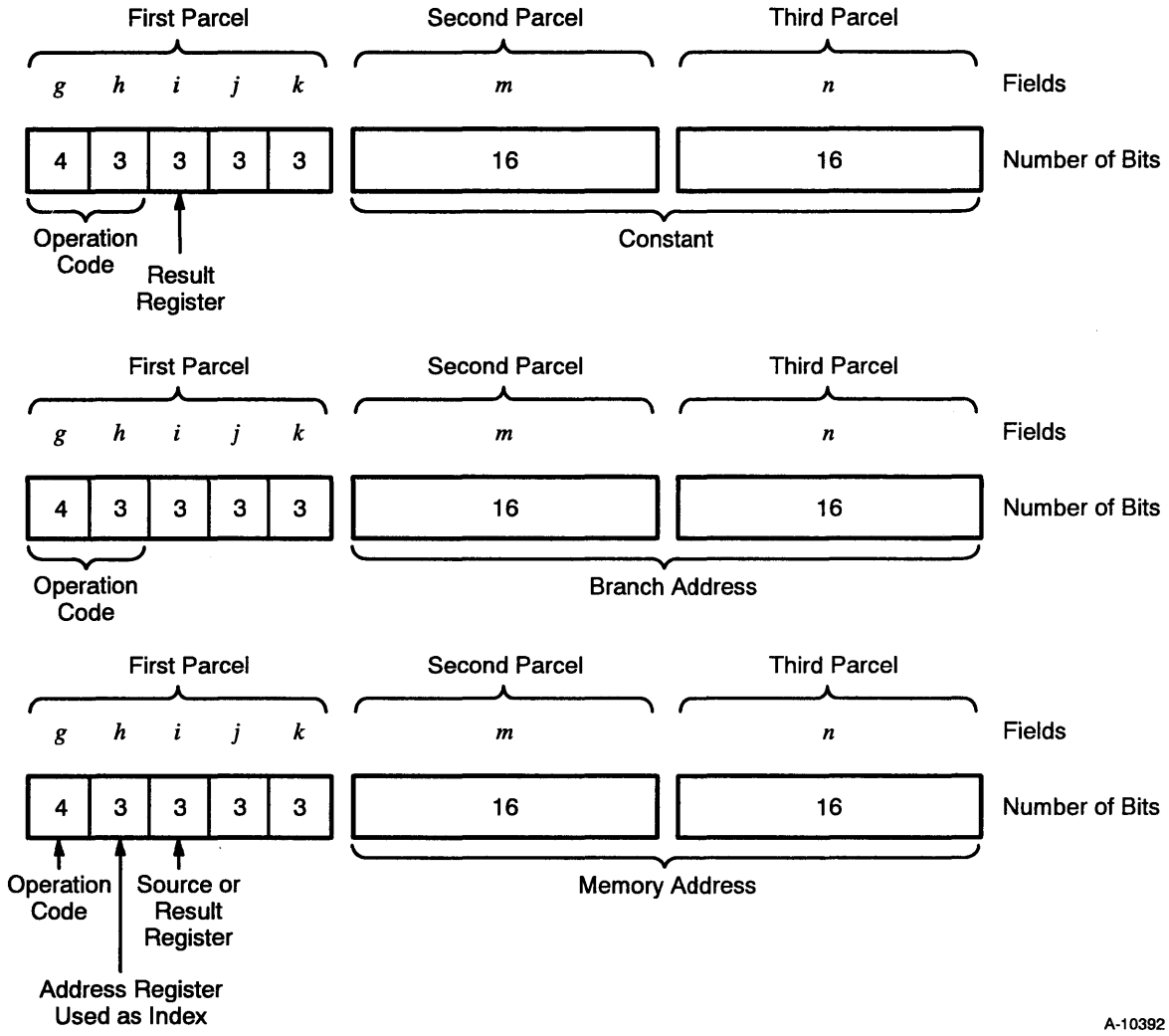


Figure 2-19. 3-parcel Instruction Format with Combined *m* and *n* Fields

Instruction Differences between Y-MP Mode and C90 Mode

The CRAY Y-MP C90 mainframe offers two modes of operation: a Y-MP compatibility mode and C90 mode. These two modes are referred to as as Y-MP mode and C90 mode, respectively. In Y-MP mode, all instructions defined within the Y-MP mode of the CRAY Y-MP computer system will function as specified for that mode. In addition, many new C90 mode instructions are supported in the Y-MP mode.

Table 2-5 lists the new instructions for the CRAY Y-MP C90 computer system or those that function differently in the CRAY Y-MP C90 and CRAY Y-MP computer systems.

Table 2-5. Instruction Comparisons

Machine Instruction	CAL Syntax	CRAY Y-MP C90 Function Description	CRAY Y-MP Function Description
001000	PASS	This is a no-operation instruction.	This is a no-operation instruction.
0012*j*2	DI,A*j*	Disable channel (A*j*) interrupts.	Disable channel (A*j*) interrupts.
0012*j*3	EI,A*j*	Enable channel (A*j*) interrupts.	Enable channel (A*j*) interrupts.
001302	EMI	Enable monitor mode interrupt modes.	Enable monitor mode interrupt modes.
001303	DMI	Disable monitor mode interrupt modes.	Disable monitor mode interrupt modes.
001406	ECI	Enable the programmable clock interrupt request.	Enable the programmable clock interrupt request.
001407	DCI	Disable the programmable clock interrupt request.	Disable the programmable clock interrupt request.
001600	ESI	Enable system I/O interrupts.	Enable system I/O interrupts.
0017*jk*	BP,*k* A*j*	Transmit (A*j*) to breakpoint address *k*.	N/A
00200*k*	VL A*k*	Transmit (A*k*) to VL register. (maximum VL = 128)	Transmit (A*k*) to VL register. (maximum VL = 64)
002301	EBP	Enable interrupt on breakpoint.	N/A
002401	DBP	Disable interrupt on breakpoint.	N/A
002704	CPA	Complete port reads and writes.	N/A
002705	CPR	Complete port reads.	N/A
002706	CPW	Complete port writes.	N/A
0030*j*1[1]	VM1 S*j*	Transmit (S*j*) to VM upper register.	N/A

[1] This instruction is supported only in C90 mode.

Table 2-5. Instruction Comparisons (continued)

Machine Instruction	CAL Syntax	CRAY Y-MP C90 Function Description	CRAY Y-MP Function Description
0034*jk*	SM,A*k* 1,TS	Test and set semaphore (A*k*). (*j*2 = 1)	N/A
0036*jk*	SM,A*k* 0	Clear semaphore (A*k*). (*j*2 = 1)	N/A
0037*jk*	SM,A*k* 1	Set semaphore (A*k*). (*j*2 = 1)	N/A
0051*jk*	Jinv B*jk*	Jump to (B*jk*). (maintenance only, invalidate IB)	N/A
006000 *nm*[1]	J *exp*	Jump to *nm*.	N/A
0064*jk nm*[1]	JTS*jk exp*	Branch to *nm* if SM*jk* = 1; else set SM*jk* = 1. (*j*2 = 0)	N/A
0064*jk nm*[1]	JTS,A*k exp*	Branch to *nm* if SM,A*k* = 1; else set SM,A*k* = 1. (*j*2 = 1)	N/A
007000 *nm*[1]	R *exp*	Return jump to *nm*; set B00 to (P) + 3.	N/A
010000 *nm*[1]	JAZ *exp*	Jump to *nm* if (A0) = 0.	N/A
011000 *nm*[1]	JAN *exp*	Jump to *nm* if (A0) ≠ 0.	N/A
012000 *nm*[1]	JAP *exp*	Jump to *nm* if (A0) ≥ 0.	N/A
013000 *nm*[1]	JAM *exp*	Jump to *nm* if (A0) < 0.	N/A
014000 *nm*[1]	JSZ *exp*	Jump to *nm* if (S0) = 0.	N/A
015000 *nm*[1]	JSN *exp*	Jump to *nm* if (S0) ≠ 0.	N/A
016000 *nm*[1]	JSP *exp*	Jump to *nm* if (S0) ≥ 0.	N/A
017000 *nm*[1]	JSM *exp*	Jump to *nm* if (S0) < 0.	N/A
023*i*01	A*i* VL	Transmit (VL) to A*i*. (maximum VL = 128)	Transmit (VL) to A*i*. (maximum VL = 64)
026*ij*4	A*i* SB,A*j*,+1	Transmit (SB) designated by (A*j*) to A*i*; increment by 1.	N/A
026*ij*5	A*i* SB*j*,+1	Transmit (SB*j*) to A*i*; increment by 1.	N/A
026*ij*6	A*i* SB,A*j*	Transmit (SB) designated by (A*j*) to A*i*.	N/A

[1] These instructions are supported only in C90 mode.

Table 2-5. Instruction Comparisons (continued)

Machine Instruction	CAL Syntax	CRAY Y-MP C90 Function Description	CRAY Y-MP Function Description
027*ij*6	SB,A*j* A*i*	Transmit (A*i*) to SB designated by (A*j*).	N/A
033*ij*1	A*i* CE,A*j*	Transmit error flag of channel (A*j*) to A*i* ($j \neq 0$); include done flag.	Transmit error flag of channel (A*j*) to A*i*. ($j \neq 0$)
040*i*20 *nm*	S*i* S*i*:*exp*	Transmit *nm* to S*i* bits 2^0–2^{31}. (bits 2^{32}–2^{63} are unchanged)	N/A
040*i*40 *nm*	S*i* *exp*:S*i*	Transmit *nm* to S*i* bits 2^{32}–2^{63}. (bits 2^0–2^{31} are unchanged)	N/A
072*ij*6	S*i* ST,A*j*	Transmit (ST) designated by (A*j*) to S*i*.	N/A
073*i*10[1]	S*i* VM1	Transmit VM1 to S*i*.	N/A
073*i*21	S*i* SR2	Read PM counters 00–17 and increment pointer.	Increment performance counter.
073*i*25	SR2 S*i*	Issue PM maintenance advance.	N/A
073*i*31	S*i* SR3	Read PM counters 20–37 and increment pointer.	Clear all maintenance modes.
073*i*75	SR7 S*i*	Transmit (S*i*) to maintenance mode register.	N/A
073*ij*1	S*i* SR*j*	Transmit (SR*j*) to S*i*.	N/A
073*ij*5	SR*j* S*i*	Transmit (S*i*) to SR*j*.	N/A
073*ij*6	ST,A*j* S*i*	Transmit (S*i*) to ST designated by (A*j*).	N/A
005400 150*ij*0	V*i* V*j* < V0	Shift (V*j*) left (V0) places to V*i*.	N/A
005400 151*ij*0	V*i* V*j* > V0	Shift (V*j*) right (V0) places to V*i*.	N/A
005400 152*ijk*	V*i* V*j*,A*k*	Transfer (V*j*) to (V*i*) starting at element (A*k*).	N/A
174*ij*3	V*i* ZV*j*	Transmit leading zero count of (V*j*) to V*i*.	N/A

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. This special operand is available regardless of existing A0 or S0 reservations, and in this case the reservation on the register is not checked. This special operand does not alter the actual value of the S0 or A0 register. If register S0 or A0 is referenced in the *i* field as an operand, the value stored in the register is used. The CAL assembler issues a caution-level error message for A0 or S0 when 0 does not apply to the *i* field. Table 2-6 lists the special register values.

Table 2-6. Special Register Values

Field	Operand Value
A*h*, *h* = 0	0
A*j*, *j* = 0	0
A*k*, *k* = 0	1
S*j*, *j* = 0	0
S*k*, *k* = 0	$2^{63} = 1$

Special CAL Syntax Forms

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 CAL instruction set.

For example, the following CAL instructions generate machine instruction 002000, which enters a 1 into the vector length (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.

Monitor Mode Instructions

The monitor mode instructions (channel control, set real-time clock, programmable clock interrupts, and so on) 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 instruction.

In several cases, a single CAL instruction 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.

Program Range

The program range, or maximum program length, is 4 Mwords in Y-MP mode and 1 Gword in C90 mode. An instruction outside these ranges produces an undefined result.

CPU Instruction Summary

This subsection introduces and summarizes all mainframe instructions used by the CRAY Y-MP C90 computer system. 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; acronyms and abbreviations are defined there.

In some instructions, register designators are prefixed by the following letters that have special meaning to the assembler. 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

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 exclusive OR of specified registers

An expression (*exp*) occupies the *jk*, *ijkm*, or *mn* 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 transmit or control operations are performed by specialized hardware components known as functional units. Listed below are the machine 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 through 051
Scalar shift	052 through 057
Scalar pop/parity/leading zero	026, 027
Vector add (integer)	154 through 157
Vector logical	140 through 147, 175
Second vector logical	140 through 145
Vector shift	150 through 153
Vector pop/parity	174*ij*1, 174*ij*2
Floating-point add	062, 063, 170 through 173
Floating-point multiply	064 through 067, 160 through 167
Floating-point reciprocal	070, 174*ij*0
Memory (scalar)	10*h* through 13*h*
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; then the machine instruction, the CAL syntax, and a description is listed.

NOTE: The following superscripts in the machine instruction column are used throughout the instruction summary:

Superscript	Description
1	C90 mode only
2	Y-MP mode only
3	Privileged to monitor mode

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
020*i*00 *nm*	A*i* *exp*	Transmit *nm* to A*i*
021*i*00 *nm*	A*i* *exp*	Transmit one's complement of *nm* to A*i*
022*ijk*	A*i* *exp*	Transmit *jk* to A*i*
031*i*00	A*i* -1	Transmit -1 to A*i*; (A*i* = 77777777)

Transfers into S Registers

The following instructions transmit values into the S registers.

Machine Instruction	CAL Syntax	Description
040*i*00 *nm*	S*i* *exp*	Transmit *nm* to S*i* 0 - 31 (32 - 63 = 0)

Machine Instruction	CAL Syntax	Description
040*i*20 *nm*	S*i* S*i*:*exp*	Transmit *nm* to S*i* 00 - 31 (32 - 63 unchanged)
040*i*40 *nm*	S*i* *exp*:S*i*	Transmit *nm* to S*i* 32 - 63 (00 - 31 unchanged)
041*i*00 *nm*	S*i* *exp*	Transmit one's complement of *nm* to S*i* (32 - 63 = 1)
042*i*00	S*i* -1	Enter -1 into S*i*; (S*i* = 177777 177777 177777 177777)
042*ijk*	S*i* <*exp*	Form ones mask *exp* bits in S*i* from the right; *jk* field contains 100_8 -*exp*
042*ijk*	S*i* #>*exp*	Form zeros mask *exp* bits in S*i* from the left; *jk* field get *exp*
042*i*77	S*i* 1	Enter 1 into S*i*
043*i*00	S*i* 0	Clear S*i*
043*ijk*	S > *exp*	Form ones mask *exp* bits in S*i* from the left; *jk* field gets *exp*
043*ijk*	S*i* #<*exp*	Form zeros mask *exp* bits in S*i* from the right; *jk* field contains 100_8 -*exp*
047*i*00	S*i* #SB	Enter one's complement of sign bit into S*i*
051*i*00	S*i* SB	Enter sign bit into S*i*
071*i*30	S*i* 0.6	Transmit constant 0.75*2**48 to S*i* (S*i* = 040060 140000 000000 000000)
071*i*40	S*i* 0.4	Transmit constant 0.4 to S*i* (S*i* = 040000 100000 000000 000000)
071*i*50	S*i* 1.0	Transmit constant 1.0 to S*i* (S*i* = 040001 100000 000000 000000)

Machine Instruction	CAL Syntax	Description
071*i*60	S*i* 2.0	Transmit constant 2.0 to S*i* (S*i* = 040002 100000 000000 000000)
071*i*70	S*i* 4.0	Transmit constant 4.0 to S*i* (S*i* = 040003 100000 000000 000000)

Transfers into V Registers

The following instructions transmit values into the V registers.

Machine Instruction	CAL Syntax	Description
077*i*0*k*	V*i*,A*k* 0	Clear V*i* element (A*k*)
145*iii*	V*i* 0	Clear V*i*

Transfers into Semaphore Registers

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* (*j*2 = 0)
0034*jk*	SM,A*k* 1,TS	Test and set semaphore (A*k*) (*j*2 = 1)
0036*jk*	SM*jk* 0	Clear semaphore *jk* (*j*2 = 0)
0036*jk*	SM,A*k* 0	Clear semaphore (A*k*) (*j*2 = 1)
0037*jk*	SM*jk* 1	Set semaphore *jk* (*j*2 = 0)
0037*jk*	SM,A*k* 1	Set semaphore (A*k*) (*j*2=1)

Interregister Transfer Instructions

The interregister 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* [VL = 128 (64)]
024*ijk*	A*i* B*jk*	Transmit (B*jk*) to A*i*
026*ij*4	A*i* SB,A*j*,+1	Transmit (SB) designated by (A*j*) to A*i*, and increment (SB,A*j*) by 1
026*ij*5	A*i* SB*j*,+1	Transmit (SB*j*) to A*i*, and increment (SB*j*) by 1
026*ij*6	A*i* SB*j*,A*j*	Transmit (SB) designated by (A*j*) to A*i*
026*ij*7	A*i* SB*j*	Transmit shared B*j* to A*i*
030*i*0*k*	A*i* A*k*	Transmit (A*k*) to A*i*
031*i*0*k*	A*i* -A*k*	Transmit the negative of (A*k*) to A*i*
033*i*00	A*i* CI	Transmit I/O interrupting channel number to A*i*
033*ij*0	A*i* CA,A*j*	Transmit I/O-current address of channel (A*j*) to A*i*
033*ij*1	A*i* CE, A*j*	Transmit channel status word (A*j*) to A*i*

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*	S*i* #S*k*	Transmit one's complement of (S*k*) to S*i*

Machine Instruction	CAL Syntax	Description
051*i*0*k*	S*i* S*k*	Transmit (S*k*) to S*i*
061*i*0*k*	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
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 (exponent equals 40060)
072*i*00	S*i* RT	Transmit (RTC) to S*i*
072*i*02	S*i* SM	Transmit (SM) to S*i*
072*ij*3	S*i* ST*j*	Transmit (ST*j*) to S*i*
073*i*00	S*i* VM	Transmit vector mask lower to S*i*
074*ijk*	S*i*T*jk*	Transmit (T*jk*) to S*i*
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*	V*i* V*k*	Transmit (V*k*) to V*i*
156*i*0*k*	V*i* -V*k*	Transmit negative of (V*k*) to V*i*

Transfers to Intermediate Registers

The following instructions transmit the contents of A and S registers into the intermediate B and T registers.

Machine Instruction	CAL Syntax	Description
025*ijk*	B*jk* A*i*	Transmit (A*i*) to B*jk*
075*ijk*	T*jk* S*i*	Transmit (S*i*) to T*jk*

Transfers to Shared Registers

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

Machine Instruction	CAL Syntax	Description
027*ij*6	SB,A*j* A*i*	Transmit (A*i*) to SB designated by (A*j*)
027*ij*7	SB*j* A*i*	Transmit (A*i*) to shared B*j*
073*i*02	SM S*i*	Transmit (S*i*) to the semaphore registers
073*ij*3	ST*j* S*i*	Transmit (S*i*) to shared T*j*
073*ij*6	ST,A*j* S*i*	Transmit (S*i*) to ST designated by (A*j*)

Transfers to Status Registers

The following instructions transfer the contents of an S register into the status registers.

Machine Instruction	CAL Syntax	Description
073*i*05	SR0 S*i*	Transmit (S*i*) to status register 0
073*i*75[3]	SR7 S*i*	Transmit (S*i*) to maintenance mode register (status register 7)

Transfer to Vector Mask Register

The following instructions transmit the contents of other registers into the vector mask register.

Machine Instruction	CAL Syntax	Description
0030*j*0	VM S*j*	Transmit (S*j*) to vector mask lower V*m* = (2127 - 264), lower elements 0 - 63
0030*j*1[1]	VM1 S*j*	Transmit (S*j*) to vector mask upper V*m* = (263 - 20), upper elements 64 - 127

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
002000	VL 1	Transmit 1 to VL

Memory Transfer Instructions

The memory transfer instructions enable or 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 (BDM = 0)
002600	EBM	Enable bidirectional memory transfers (BDM = 1)

Memory References

The following instructions ensure completion of instructions for bidirectional memory transfers.

Machine Instruction	CAL Syntax	Description
002700	CMR	Complete memory references
002704	CPA	Complete port reads and writes
002705	CPR	Complete port reads
002706	CPW	Complete port writes

The following instructions write values into memory.

Machine Instruction	CAL Syntax	Description
035*ijk*	,A0 B*jk*,A*i*	Write (A*i*) words from B register *jk* to memory address ((A0) + (DBA))
037*ijk*	,A0 T*jk*,A*i*	Write (A*i*) words from T register *jk* to memory address ((A0) + (DBA))
11*hi*00 *nm*	*exp*,A*h* A*i*	Write (A*i*) to memory address ((A*h*) + *exp* + (DBA))
13*hi*00 *nm*	*exp*,A*h* S*i*	Write (S*i*) to memory address ((A*h*) + *exp* + (DBA))
1770*jk*	,A0,A*k* V*j*	Write (VL) words from V*j* to memory address ((A0) + (DBA)) incremented by (A*k*)
1770*j*0	,A0, 1 V*j*	Write (VL) words from V*j* to memory address ((A0) + (DBA)) incremented by 1
1771*jk*	,A0,V*k* V*j*	Write (VL) words from V*j* to memory address ((A0) + (V*k*) + (DBA)) (scatter)

Reads

The following instructions read values from memory.

Machine Instruction	CAL Syntax	Description
034*ijk*	B*jk*,A*i* ,A0	Read (A*i*) words to B register *jk* from memory address ((A0) + (DBA)) Port A
036*ijk*	T*jk*,A*i* ,A0	Read (A*i*) words to T register *jk* from memory address ((A0) + (DBA)) Port B
10*hi*00 *nm*	A*i* *exp*,A*h*	Read from memory ((A*h*) + *exp* + (DBA)) to A*i*
12*hi*00 *nm*	S*i* *exp*,A*h*	Read from memory address ((A*h*) + *exp* + (DBA)) to S*i*
176*i*0*k*	V*i* ,A0,A*k*	Read (VL) words to V*i* from memory address ((A0) + (DBA)) incremented by (A*k*)
176*i*00	V*i* ,A0,1	Read (VL) words to V*i* from memory address ((A0 + (DBA)) incremented by 1
176*i*1*k*	V*i* ,A0,V*k*	Read (VL) words to V*i* from memory address ((A0 + (V*k*) + (DBA)) (gather)

Integer Arithmetic Instructions

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

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 is two's complement and is represented as such in the registers. The address add and address multiply functional units perform 36-bit (Y-MP mode) and 64-bit (C90 mode) arithmetic. The scalar add functional unit and the vector add functional unit perform 64-bit arithmetic. No overflow conditions are 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.

32-bit Integer Arithmetic

The following instructions perform 32-bit integer arithmetic.

Machine Instruction	CAL Syntax	Description
030*ijk*	A*i* A*j* + A*k*	Integer sum of (A*j*) and (A*k*) to A*i*
030*ij*0	A*i* A*j* + 1	Integer sum of (A*j*) and 1 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	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*) to V*i*
155*ijk*	V*i* V*j* + V*k*	Integer sums of (V*j*) and (V*k*) to V*i*
156*ijk*	V*i* S*j* – V*k*	Integer differences of (S*j*) and (V*k*) to V*i*
157*ijk*	V*i* V*j* – V*k*	Integer differences of (V*j*) and (V*k*) to V*i*

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 FP Interrupt (IFP = 1, Clear FPS)
002200	DFI	Disable FP Interrupt (IFP = 0, Clear FPS)

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*	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*	S*i* -FS*k*	Transmit 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*) to V*i*

Machine Instruction	CAL Syntax	Description
170*i*0*k*	V*i* +FV*k*	Normalize (V*k*) to V*i*
171*ijk*	V*i* V*j* + FV*k*	Floating-point sums of (V*j*) and (V*k*) to V*i*
172*ijk*	V*i* S*j* – FV*k*	Floating-point differences of (S*j*) and (V*k*) to V*i*
172*i*0*k*	V*i* -FV*k*	Transmit normalized negatives of (V*k*) to V*i*
173*ijk*	V*i* V*j* – FV*k*	Floating-point differences of (V*j*) and (V*k*) to V*i*

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*	Full-precision 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*) to V*i*
161*ijk*	V*i* V*j* * FV*k*	Floating-point products of (V*j*) and (V*k*) to V*i*
162*ijk*	V*i* S*j* * HV*k*	Half-precision rounded floating-point products of (S*j*) and (V*k*) to V*i*
163*ijk*	V*i* V*J* * HV*k*	Half-precision rounded floating-point products of (V*j*) and (V*k*) to V*i*

Machine Instruction	CAL Syntax	Description
164*ijk*	V*i* S*j* * RV*k*	Rounded floating-point products of (S*j*) and (V*k*) to V*i*
165*ijk*	V*i* V*j* * RV*k*	Rounded floating-point products of (V*j*) and (V*k*) to V*i*

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 approximations of (V*j*) to V*i*

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 exclusive ORs, logical equivalence, vector mask, and merges. Logical operations are defined below.

- A logical product (& operator) is the AND function.
- A logical exclusive or (\ operator) is the exclusive OR function.
- A logical sum (! operator) is the inclusive OR function.
- A logical merge combines two operands depending on a ones mask in a third operand. The result is defined by (operand 2 & mask) ! (operand 1 & #mask).

Logical Products

The following instructions peform logical product operations.

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	S*i* S*j*&SB	Sign bit of (S*j*) to S*i*
044*ij*0	S*i* SB&S*j*	Sign bit of (S*j*) to S*i* (*j* p 0)
045*ijk*	S*i* #S*k*&S*j*	Logical product of (S*j*) and one's complement of (S*k*) to S*i*
045*ij*0	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*) to V*i*
141*ijk*	V*i* V*j*&V*k*	Logical products of (V*j*) and (V*k*) to V*i*

Logical Sums

The following instructions perform logical sum operations.

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	S*i* S*j*!SB	Logical sum of (S*j*) and sign bit to S*i*
051*ij*0	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*) to V*i*
143*ijk*	V*i* V*j*!V*k*	Logical sums of (V*j*) and (V*k*) to V*i*

Logical Exclusive ORs

The following instructions perform exclusive OR operations.

Machine Instruction	CAL Syntax	Description
046*ijk*	S*i* S*j*\S*k*	Exclusive OR of (S*j*) and (S*k*) to S*i*
046*ij*0	S*i* S*j*\SB	Toggle sign bit of (S*j*), then enter into S*i*
046*ij*0	S*i* SB\S*j*	Toggle sign bit of (S*j*), then enter into S*i* ($j \neq 0$)
144*ijk*	V*i* S*j*\V*k*	Exclusive ORs of (S*j*) and (V*k*) to V*i*
145*ijk*	V*i* V*j*\V*k*	Exclusive ORs of (V*j*) and (V*k*) to V*i*

Logical Equivalence

The following instructions perform logical equivalence operations.

Machine Instruction	CAL Syntax	Description
047*ijk*	S*i* #S*j*\S*k*	Logical equivalence of (S*k*) and (S*j*) to S*i*
047*ij*0	S*i* #SB\S*j*	Logical equivalence of (S*j*) and sign bit to S*i* ($j \neq 0$)
047ij0	S*i* #S*j*\SB	Logical equivalence of (S*j*) and sign bit to S*i*

Vector Mask

The following instructions test the elements of a vector register and use the test results to set the corresponding bits of the vector mask register.

Machine Instruction	CAL Syntax	Description
1750*j*0	VM Vj,Z	Set VM bit if (V*j*) = 0
1750*j*1	VM Vj,N	Set VM bit if (V*j*) ≠ 0
1750*j*2	VM Vj,P	Set VM bit if (V*j*) ≥ 0
1750*j*3	VM Vj,M	Set VM bit if (V*j*) < 0
175*ij*4	V*i*,VM Vj,Z	Set VM bit if (V*j*) = 0; also, store the compressed indices of the V*j* elements = 0 in the V*i* elements
175*ij*5	V*i*,VM Vj,N	Set VM bit if (V*j*) ≠ 0; also, store the compressed indices of the V*j* elements ≠ 0 in the V*i* elements
175*ij*6	V*i*,VM Vj,P	Set VM bit if (V*j*) ≥ 0; also, store the compressed indices of the V*j* elements ≥ 0 in the V*i* elements
175*ij*7	V*i*,VM Vj,M	Set VM bit if (V*j*) < 0; also store the compressed indices of the V*j* elements < 0 in the V*i* elements

Merge

The following instructions perform a logical merge that combines two operands according to the bits set in a ones mask in a third operand.

Machine Instruction	CAL Syntax	Description
050*ijk*	S*i* Sj!Si&Sk	Logical product of (S*i*) and (S*k*) complement ORed with logical product of (S*j*) and (S*k*) to S*i*
050*ij*0	S*i* Sj!Si&SB	Scalar merge of (S*i*) and sign of (S*j*) to S*i*

Machine Instruction	CAL Syntax	Description
146*ijk*	V*i* S*j*!V*k*&VM	Transmit (S*j*) if VM bit = 1 or (V*k*) if VM bit = 0 to V*i*
146*i*0*k*	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 or (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* = *jk* places to S0
053*ijk*	S0 S*i* > *exp*	Shift (S*i*) right *exp* = 100_8 - *jk* places to S0
054*ijk*	S*i* S*i* < *exp*	Shift (S*i*) left *exp* = *jk* places to S*i*
055*ijk*	S*i* S*i* > *exp*	Shift (S*i*) right *exp* = 100_8 - *jk* places to S*i*
056*ijk*	S*i* S*i*,S*j* < A*k*	Shift (S*i*) and (S*j*) left (A*k*) places to S*i*
056*ij*0	S*i* S*i*,S*j*<1	Shift (S*i*) and (S*j*) left one place to S*i*
056*i*0*k*	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 (A*k*) places to S*i*
057*ij*0	S*i* S*j*,S*i*>1	Shift (S*j*) and (S*i*) right one place to S*i*

Machine Instruction	CAL Syntax	Description
057*i*0*k*	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*) left (A*k*) places to V*i*
150*ij*0	V*i* V*j* < 1	Shift (V*j*) left one place to V*i*
005400 150*ij*0	V*i* V*j* < V0	Shift (V*j*) left (V0) places to V*i*
151*ijk*	V*i* V*j* > A*k*	Shift (V*j*) right (A*k*) places to V*i*
151*ij*0	V*i* V*j* > 1	Shift (V*j*) right one place to V*i*
005400 151*ij*0	V*i* V*j* > V0	Shift (V*j*) right (V0) places to V*i*
152*ijk*	V*i* V*j*,V*j*< A*k*	Double shift (V*j*) left (A*k*) places to V*i*
152*ij*0	V*i* V*j*,V*j*<1	Double shift (V*j*) left one place to V*i*
005400 152*ijk*	V*i* V*j*,A*k*	Vector word shift of (V*j*) starting at element (A*k*) to V*i* ((A*k*)<VL)
153*ijk*	V*i* V*j*,V*j* > A*k*	Double shift (V*j*) right (A*k*) places to V*i*
153*ij*0	V*i* V*j*,V*j*>1	Double shift (V*j*) right one place to V*i*

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 counts of (V*j*) to V*i*

Population Parity Count

The following instructions perform population parity count.

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

Scalar Leading Zero Count

The following instruction performs the scalar leading zero count.

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

Vector Leading Zero Count

The following instruction performs the vector leading zero count.

Machine Instruction	CAL Syntax	Description
174*ij*3	V*i* ZV*j*	Leading zero count of (V*j*) to V*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*)
0051*jk*	Jinv B*jk*	Jump to (B*jk*) (Maintenance only: invalidates instruction buffers)
006*ijkm*[2]	J *exp*	Jump to *exp*
006000 *nm*[1]	J *exp*	Jump to *exp*

Conditional Branch Instructions

The following instructions perform conditional branch operations.

Machine Instruction	CAL Syntax	Description
0064*jk nm*[1]	JTS*jk* *exp*	Branch to *exp* if (SM*jk*) = 1; else set SM*jk* (*j*2 = 0)
0064*jk nm*[1]	JTS,A*k* *exp*	Branch to *exp* if (SM,(A*k*)) = 1; else set SM,(A*k*) (*j*2 = 1)
010*ijkm*[2]	JAZ *exp*	Jump to *exp* if (A0) = 0 (*i*2 = 0)
010000 *nm*[1]	JAZ *exp*	Jump to *exp* if (A0) = 0
011*ijkm*[2]	JAN *exp*	Jump to *exp* if (A0) ≠ 0 (*i*2 = 0)
011000 *nm*[1]	JAN *exp*	Jump to *exp* if (A0) ≠ 0
012*ijkm*[2]	JAP *exp*	Jump to *exp* if (A0) is positive; (A0) ≥ 0 (*i*2 = 0)
012000 *nm*[1]	JAP *exp*	Jump to *exp* if (A0) is positive; (A0) ≥ 0
013*ijkm*[2]	JAM *exp*	Jump to *exp* if (A0) is negative (*i*2 = 0)
013000 *nm*[1]	JAM *exp*	Jump to *exp* if (A0) is negative
014*ijkm*[2]	JSZ *exp*	Jump to *exp* if (S0) = 0 (*i*2 = 0)

Machine Instruction	CAL Syntax	Description
014000 *nm*[1]	JSZ *exp*	Jump to *exp* if (S0) = 0
015*ijkm*[2]	JSN *exp*	Jump to *exp* if (S0) ≠ 0 (*i*2 = 0)
015000 *nm*[1]	JSN *exp*	Jump to *exp* if (S0) ≠ 0
016*ijkm*[2]	JSP *exp*	Jump to *exp* if (S0) is positive; (S0) ≥ 0 (*i*2 = 0)
016000 *nm*[1]	JSP *exp*	Jump to *exp* if (S0) is positive; (S0) ≥ 0
017*ijkm*[2]	JSM *exp*	Jump to *exp* if (S0) is negative (*i*2 = 0)
017000 *nm*[1]	JSM *exp*	Jump to *exp* if (S0) is negative

Return Jump

The following instructions perform a return jump operation.

Machine Instruction	CAL Syntax	Description
007*ijkm*[2]	R *exp*	Return jump to *exp* and set register B00 to (P) + 2
007000 *nm*[1]	R *exp*	Return jump to *exp* and set register B00 to (P) + 3

Normal Exit

The following instruction performs a normal exit operation.

Machine Instruction	CAL Syntax	Description
004000	EX	Normal Exit

Error Exit

The following instruction performs an error exit operation.

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*[3]	CA,A*j* A*k*	Set channel (A*j*) CA register to (A*k*) and begin I/O sequence
001000[3]	NOP	Pass
0011*jk*[3]	CL,A*j*	Set channel (A*j*) CL register to (A*k*)
0012*j*0[3]	CI,A*j*	Clear channel (A*j*) interrupt and error flags; clear device master clear (output channel)
0012*j*1[3]	MC,A*j*	Clear channel (A*j*) interrupt and error flags, set device master clear (output channel); clear device ready-held (input channel)

Set Exchange Address

The following instruction sets the exchange address.

Machine Instruction	CAL Syntax	Description
0013*j*0[3]	XA A*j*	Enter XA register with (A*j*)

Set Real-time Clock

The following instruction performs a real-time clock operation.

Machine Instruction	CAL Syntax	Description
0014*j*0[3]	RT S*j*	Enter RTC register with (S*j*)

Set Cluster Number

The following instruction sets the cluster number.

Machine Instruction	CAL Syntax	Description
0014*j*3[3]	CLN A*j*	Transmit (A*j*) to cluster number

Programmable Clock Interrupt

The following instructions perform programmable clock operations.

Machine Instruction	CAL Syntax	Description
0014*j*4[3]	PCI S*j*	Transmit (S*j*) to programmable clock
001405[3]	CCI	Clear programmable clock interrupt (PCI) request
001406[3]	ECI	Enable PCI (MM IPC only)
001407[3]	DCI	Disable PCI (MM IPC only)

Operand Range Error Interrupt

The following instructions enable or disable operand range error interrupts.

Machine Instruction	CAL Syntax	Description
002300	ERI	Enable range interrupt (IOR = 1)
002400	DRI	Disable range interrupt (IOR = 0)

Interprocessor Interrupt

The following instructions perform interprocessor interrupt operations.

Machine Instruction	CAL Syntax	Description
0014*j*1[3]	SIPI A*j*	Set interprocessor interrupt of CPU (A*j*)
001401[3]	SIPI	Send interprocessor interrupt to CPU0
001402[3]	CIPI	Clear interprocessor interrupt

Breakpoint Interrupt

The following instructions enable or disable breakpoint interrupts.

Machine Instruction	CAL Syntax	Description
002301	EBP	Enable breakpoint interrupt (IBP = 1)
002401	DBP	Disable breakpoint interrupt (IBP = 0)

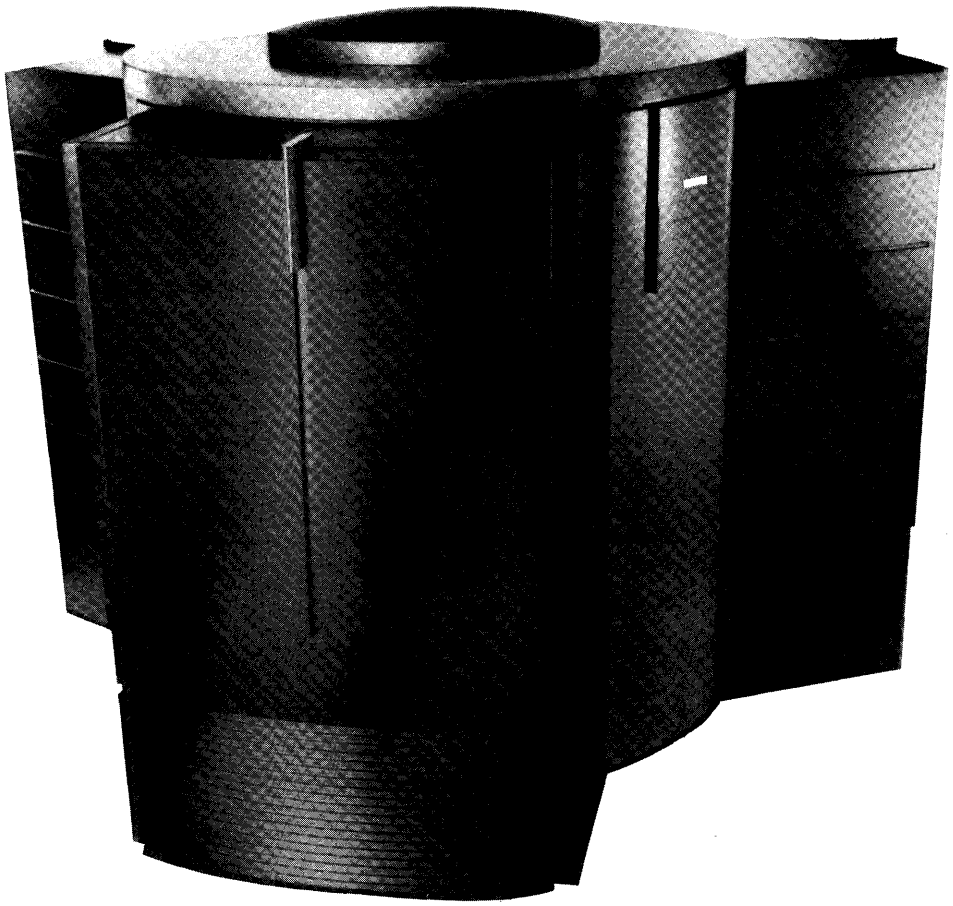
Performance Counters

The following instructions operate the performance monitor.

Machine Instruction	CAL Syntax	Description
001500[3]		Clear all performance monitor Counters
073*i*21[3]	S*i* SR2	Read PM counters 00 - 17 and increment pointer
073*i*25[3]	SR2 S*i*	Issue PM maintenance advance
073*i*31[3]	S*i* SR3	Read PM counters 20 - 37 and increment pointer

CRAY Y-MP C90 Mainframe Specifications

Model Number
CRAY Y-MP C90/16256



CRAY RESEARCH, INC.

CRAY Y-MP C90
Mainframe Specifications

System Clock

Speed ..... 4.2 ns

CPU Specifications

Number of CPUs 8, 16

Number of registers per CPU:
- Address (A) registers 8
  32 bits each
- Intermediate address (B) registers 64
  32 bits each
- Scalar (S) registers 8
  64 bits each
- Intermediate scalar (T) registers 64
  64 bits each
- Vector (V) registers 8
  64 bits x 128 elements
  (C90 mode)
  64 bits x 64 elements
  (Y-MP mode)
- Vector length (VL) register 1
  8 bits (C90 mode)
  7 bits (Y-MP mode)
- Vector mask (VM, VM1) registers 2
  64-bits each
- Program address (P) register 1
  32 bits (C90 mode)
  24 bits (Y-MP mode)

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 2
- Vector shift 2
- Vector population/parity/leading zero .. 2
- Full vector logical 2
- 2nd vector logical 2
- Vector population/parity/leading zero .. 2
- Floating-point addition 2
- Floating-point multiplication 2
- Floating-point reciprocal approx. 2

Shared Resources

Input/output section:
2 Input/output subsystems model E (IOS-E)
- 6- or 20-Mbyte/s LOSP channels 16
- 200-Mbytes/s HISP channels 16
- 1800-Mbyte/s VHISP channels 4

Central memory:
- Word width ..... 64 bits
- SBCDBD error correction ..... 16 bits
- Memory size ..... 256 Mwords
- Number of banks 1,024
- Number of modules 16
- Number of ports per CPU 4

Number of clusters 17

Number of shared registers contained in each cluster:
- Shared address (SB) registers 8
  32 bits each
- Shared scalar (ST) registers 8
  64 bits each
- Semaphore (SM) registers 32
  1 bit each

Real-time clock (64-bits) 1

Physical Description

Height ..... 85.5 (217 cm)
Width ..... 116.25 in. (295 cm)
Depth ..... 101.00 in. (257 cm)
Weight ..... 16,616 lbs (7,536 kg)

Support Equipment

Operator workstation 1
Maintenance workstation 1
Heat exchanger units 1
Refrigeration and condensing unit 1
Motor-generator sets ..... 1 or 2

3 I/O SUBSYSTEM

The Cray Research input/output subsystem model E (IOS-E) controls data transfers between several components of the CRAY Y-MP C90 computer system. The IOS-E transfers data to and receives data from the following components.

- The CRAY Y-MP C90 mainframe
- The SSD solid-state storage device model E (SSD-E)
- Peripheral devices such as disk drives and front-end computers
- The operator workstation model E (OWS-E)
- The maintenance workstation model E (MWS-E)

The IOS-E comprises a maximum of eight clusters and two workstation interfaces (WINs). The following subsections describe the I/O clusters and WINs. A block diagram of an I/O cluster is provided in Figure 3-1.

I/O Cluster

Each I/O cluster contains the following components:

- 1 I/O processor multiplexer (IOP MUX)
- 4 auxiliary I/O processors (EIOPs)
- 16 I/O buffers
- 1 low-speed (LOSP) channel pair
- 2 high-speed (HISP) channel pairs
- 16 channel adapters

The IOPs (IOP MUX and EIOPs) provide internal control for the I/O cluster. The LOSP channels allow the I/O cluster to exchange control information with the mainframe. The I/O buffers, HISP channels, and channel adapters provide the paths for data transfers between the IOS-E and the mainframe, SSD-E, and peripheral devices.

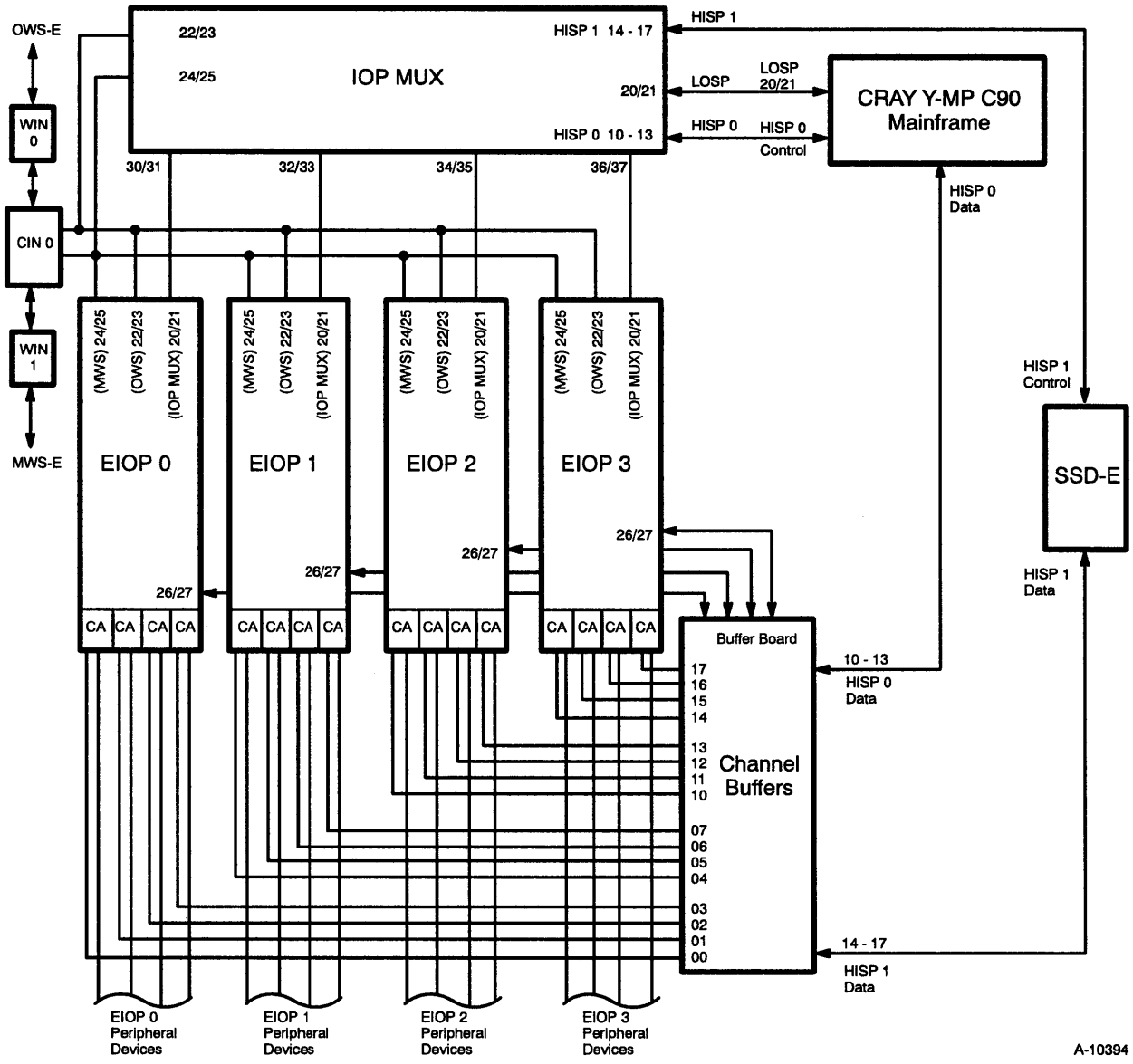


Figure 3-1. Cluster 0 Channel Connections

The following subsections describe the components of an I/O cluster.

I/O Processor

The IOPs control all data transfers in to and out of the I/O cluster. The IOP MUX communicates with the mainframe and controls data transfers to or from the mainframe. The IOP MUX also controls data transfers to or from the SSD-E. The four EIOPs control data transfers to or from peripheral devices. Each EIOP can communicate with the IOP MUX but not with other EIOPs.

The IOPs are identical; they have the same architecture and execute the same instruction set. Each IOP is a high-speed 16-bit (1 parcel) computer designed to efficiently control data transfers. Each IOP contains a 64-Kparcel local memory that is protected with SECDED (single-error correction/double-error detection) logic. Each IOP also contains a 128-parcel operand register file that is parity protected and three programmable registers. Each IOP contains 29 I/O channels; 5 of the channels monitor and control operations within the IOP, and 24 channels enable the IOP to communicate with external devices.

Each IOP executes a set of 128 1-parcel or 2-parcel instructions. Ninety-six instructions perform basic operations such as data transfers; arithmetic (two's complement), logical, and shift operations; conditional and unconditional jumps; and subroutine calls and exits. Thirty-two instructions, called I/O functions, control and monitor the I/O channels.

I/O Buffers

The 16 I/O buffers provide temporary storage for data transferred to or from the IOS-E. Each buffer contains 64 Kwords. Each word is 64 bits long and is protected with SECDED logic. Each buffer can simultaneously pass data to or from a channel adapter while passing data to or from the mainframe or SSD-E. Each buffer is dedicated to one peripheral device, or in the case of mass storage devices, to one group of identical devices.

Each EIOP is dedicated to 4 of the 16 I/O buffers. Each EIOP controls all transfers between its buffers and peripheral devices. The EIOPs work with the IOP MUX to control transfers between the buffer and the mainframe or SSD-E. Each EIOP can also transfer data between its I/O buffers and its local memory.

Low-speed and High-speed Channels

The LOSP and HISP channel pairs enable the I/O cluster to communicate with the mainframe and SSD-E. The LOSP channel pair transfers control information between the IOP MUX and the mainframe. One HISP channel pair transfers data between the I/O buffers and the mainframe; the second pair transfers data between the I/O buffers and the SSD-E.

The LOSP channel pair operates at 6 or 20 Mbytes/s. The LOSP channels are 16 bits wide and contain 4 parity bits for error detection. Each channel can operate simultaneously.

The HISP channel pairs operate at 200 Mbytes/s. The HISP channels are 64 bits wide and contain 8 SECDED bits. Each channel can operate simultaneously, but each must use a different I/O buffer.

Channel Adapters

Channel adapters enable the I/O cluster to communicate with peripheral devices. Several types of channel adapters are available; each type of channel adapter enables the I/O cluster to communicate with a different type of device. Most channel adapters can be connected to only one peripheral device; however, channel adapters for disk storage devices can be connected to multiple devices of the same type.

Each channel adapter corresponds to one I/O buffer. During a data transfer from a peripheral device, the channel adapter converts the input data from the device's format to 64-bit words, generates SECDED bits, and then transmits the data and SECDED bits to the I/O buffer. During a data transfer to an external device, the channel adapter receives 64-bit data words (plus SECDED bits) from the I/O buffer, converts the data to the correct format for that device, and then transmits the data to the device.

Each EIOP controls four channel adapters. The EIOPs control and monitor all data transfers between the I/O buffers and peripheral devices.

The following subsections provide specific information on each type of channel adapter. The specification sheet at the end of this section provides a quick reference summary of all channel adapter specifications.

CCA-1 Channel Adapter

The CCA-1 channel adapter contains a LOSP channel pair that transfers data between an I/O buffer and an external device such as a front-end interface. The LOSP channel pair consists of an input and an output channel. Both channels can operate simultaneously.

Each CCA-1 can support one external device. The maximum transfer rate between the CCA-1 and the external device is 6 or 12 Mbytes/s (software controlled). The word width of each transfer is 16 bits.

All data transfers to or from the CCA-1 are checked for data errors. Data transfers between the CCA-1 and either the mainframe or the SSD-E are protected by SECDED. Data transfers between the CCA-1 and peripheral devices are checked for parity errors.

DCA-1 Channel Adapter

The DCA-1 disk channel adapter transfers data between the I/O buffer and a disk drive. The DCA-1 disk channel adapter is compatible with the DD-40, DD-41, and DD-49 disk drives.

Each DCA-1 can support one DD-49 disk drive, two DD-40 disk drives, or two DD-41 disk drives. The maximum transfer rate between the DCA-1 and the disk drive is 12 Mbytes/s. The word width of each transfer is 16 bits.

All data transfers to or from the DCA-1 are checked for data errors. Data transfers between the DCA-1 and either the mainframe or the SSD-E are protected by SECDED. Data transfers between the DCA-1 and disk drives are checked for parity errors.

DCA-2 Channel Adapter

The DCA-2 disk channel adapter transfers data between the I/O buffer and a disk drive. The DCA-2 disk channel adapter is compatible with DD-60 and DD-61 disk drives.

Each DCA-2 can support up to eight DD-60 or DD-61 disk drives. The maximum transfer rate between the DCA-2 and the disk drive is 24 Mbytes/s. The word width of each transfer is 16 bits.

All data transfers to or from the DCA-2 are checked for data errors. Data transfers between the DCA-2 and either the mainframe or the SSD-E are protected by SECDED. Data transfers between the DCA-2 and disk drives are checked for parity errors and cyclical redundancy check (CRC) errors.

HCA-3 and HCA-4 Channel Adapters

The HCA-3 and HCA-4 channel adapters enable the IOS-E to communicate with external devices that use a High Performance Parallel Interface (HIPPI) channel. The HIPPI channel pair consists of an input and output channel. Refer to the "High Performance Parallel Interface (HIPPI)" subsection in Section 5 of this manual for more information on the HIPPI channel. The input channel is connected to HCA-3; the output channel is connected to HCA-4.

The HIPPI channel can provide high-speed communications between Cray Research computer systems. The HIPPI channel also enables a Cray Research computer system to be connected to peripheral equipment such as network adapters and graphic display devices.

Each HCA-3 or HCA-4 channel adapter can support one external HIPPI device. The maximum transfer rate between the HCA-3 or HCA-4 and the external device is 100 Mbytes/s. The word width of each transfer is 32 bits.

All data transfers to or from the HCA-3 are checked for data errors. Data transfers from the peripheral equipment to the HCA-3 are checked for parity errors and length/longitudinal redundancy check (LLRC) errors. Data transfers from the HCA-3 to the mainframe or SSD-E are protected by SECDED.

All data transfers to or from the HCA-4 are checked for data errors. Data transfers from the mainframe or the SSD-E to the HCA-4 are protected by SECDED. Data transfers from the HCA-4 to the peripheral equipment are checked for parity errors and LLRC errors.

TCA-1 Channel Adapter

The TCA-1 tape channel adapter transfers data between the I/O buffer and tape controllers. The TCA-1 channel adapter supports IBM compatible tape controllers.

Each TCA-1 can support up to eight controllers. The maximum number of tape drives each controller supports varies with different tape drive models. The maximum transfer rate between the TCA-1 and a tape drive will also vary with different tape drive models. The word width of each transfer is 8 bits.

All data transfers to or from the TCA-1 are checked for data errors. Data transfers between the TCA-1 and either the mainframe or the SSD-E are protected by SECDED. Data transfers between the TCA-1 and peripheral devices are checked for parity errors.

Workstation Interfaces

The two WINs communicate with the workstations: one WIN connects to the operator workstation; the second WIN connects to the maintenance workstation. Each WIN has a 6-Mbyte/s channel pair; each pair consists of an input and an output channel. Each channel contains 16 data bits and 4 parity bits for data error detection.

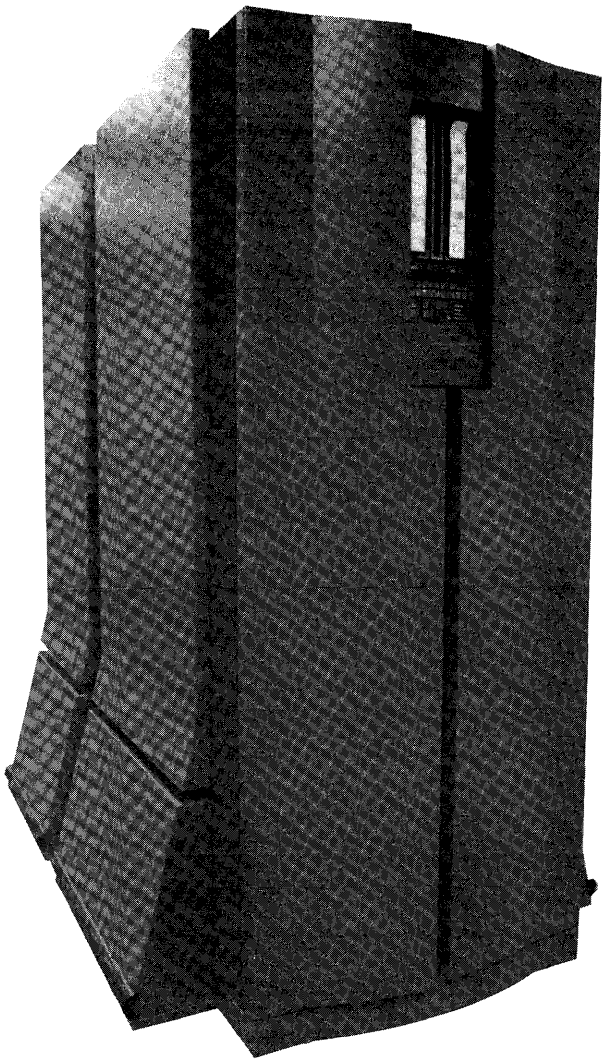
The workstations communicate with the entire IOS-E through the WINs. Each workstation can send WIN commands that affect the entire IOS-E, a single I/O cluster, or a single I/O processor. The workstations can master clear the entire IOS-E, an individual I/O cluster, or an individual I/O processor. They can also transfer data to or from any IOP, deadstart an IOP, and monitor IOPs for errors.

The IOPs can send requests to the WINs to transfer data to or from a workstation. However, a workstation receiving a request must send a specific command to the requesting IOP before the transfer can begin.

PINT

The Programmable Interrupt (PINT) enables any IOC in the IOS-E to interrupt any CPU in the mainframe. The PINT receives a 16-bit Programmed Interrupt signal from each IOC. Each bit in the Programmed Interrupt signal corresponds to a CPU within the mainframe. For example, setting bit 2^1 interrupts CPU 1 in the mainframe.

IOS Model E Specifications



CRAY RESEARCH, INC.

IOS-E Specifications

General Specifications

I/O clusters ..... 2 to 8
Workstation interfaces 2
Clock speed ..... 160 MHz (6.25 ns)

I/O Cluster

I/O processors:
- IOP MUX 1
- EIOP 4
- Word width ..... 16 bits (1 parcel)
- Local memory size ..... 64 Kparcels

I/O buffers 16
- Word width ..... 64 bits
- Size ..... 64 Kwords
- ..... (256 Kwords in development)

I/O channels (CPU connection):
- Low-speed channel pair 1
  - Operation ..... full duplex
  - Channel width ..... 16 bits (1parcel)
  - Transfer rate ..... 6 Mbytes/s
  - Data protection ..... parity
- High-speed channel pairs 2
  - Operation ..... full duplex
  - Channel width ..... 64 bits
  - Transfer rate ..... 200 Mbytes/s
  - Data protection ..... SECDED
- Channel adapters 16

Channel Adapters

CCA-1:
- Operation ..... full duplex
- Word width
  - I/O buffer side ..... 64 bits
  - External device side ..... 16 bits
- Transfer rate ..... 6 or 12 Mbytes/s
- Data protection
  - Mainframe to CCA-1 .... SECDED
  - CCA-1 to external interface .. parity
- Associated peripheral devices ..... front-end interfaces
- Maximum number of peripheral devices per CCA-1 1

Channel Adapters (continued)

DCA-1:
- Operation ..... half duplex
- Word width:
  - I/O buffer side ..... 64 bits
  - External device side ..... 16 bits
- Transfer rate ..... 12 Mbytes/s
- Data protection:
  - Mainframe to DCA-1 .... SECDED
  - DCA-1 to external interface .. parity
- Associated peripheral devices ..... DD-40, DD-41 and DD-49 disk drives
- Maximum number of peripheral devices per DCA-1 ..... one DD-49, two DD-40s, or two DD-41s

DCA-2:
- Operation ..... half duplex
- Word width:
  - I/O buffer side ..... 64 bits
  - External device side ..... 16 bits
- Transfer rate ..... 24 Mbytes/s
- Data protection:
  - Mainframe to DCA-2 .... SECDED
  - DCA-2 to external interface ..... parity and CRC
- Associated peripheral devices DD-60 and DD-61 disk drives
- Maximum number or peripheral devices per DCA-2 ..... eight DD-60s or eight DD-61s

HCA-3 (HIPPI input channel):
- Operation ..... simple duplex
- Word width:
  - I/O buffer side ..... 64 bits
  - External device side ..... 32 bits
- Transfer rate ..... 100 Mbytes/s
- Data protection:
  - Mainframe to HCA-3 .... SECDED
  - HCA-3 to external interface .. parity
- Associated peripheral devices ..... All HIPPI devices
- Maximum number of peripheral devices per HCA-3 1

IOS-E Specifications

Channel Adapters (continued)

HCA-4 (HIPPI output channel):
- Operation ..... simple duplex
- Word width:
  - I/O buffer side ..... 64 bits
  - External device side ..... 32 bits
- Transfer rate ..... 100 Mbytes/s
- Data protection:
  - Mainframe to HCA-4 .... SECDED
  - HCA-4 to external interface .. parity
- Associated peripheral devices ..... all HIPPI devices
- Maximum number of peripheral devices per HCA-4 1

TCA-1:
- Operation ..... simple duplex
- Word width:
  - I/O buffer side ..... 64 bits
  - External device side ..... 8 bits
- Transfer rate ..... 1.5 to 4.5 Mbytes/s
- Data protection:
  - Mainframe to TCA-1 .... SECDED
  - SSD to TCA-1 ..... SECDED
  - TCA-1 to external device .... parity
- Associated peripheral devices ..... IBM compatible tapes
- Maximum number of tape controllers per TCA-1 8

Physical Description of an IOS Chassis

- Height ..... 76.25 in. (194 cm)
- Width ..... 46 in. (117 cm)
- Depth ..... 73.5 in. (187 cm)
- Weight ..... 7,695 lbs (3,182 kgs)
- Floor loading .. 520 lbs/ft^2 (2,538 kg/m^2)
- Access requirements ..... 3 ft (0.9 m) on all sides
- Heat dissipation to air ..... 3.15 kW (maximum)

4 SSD SOLID-STATE STORAGE DEVICE

The Cray Research SSD solid-state storage device model E (SSD-E) is an optional high-performance device used for temporary data storage. The SSD-E functions like a disk drive. Because of its fast access time, fast transfer rates, and large storage capacity, the SSD-E enhances the performance of a Cray Research computer system by significantly reducing I/O processing time. The storage medium in an SSD-E is solid-state, dynamic random access memory (DRAM) chips rather than magnetic film. The transfer rates to and from the SSD-E are considerably faster than that of a disk drive. Data sets for the SSD-E are identical to data sets for disk drives, providing portability and flexibility.

The mainframe, I/O subsystem model E (IOS-E), and maintenance workstation model E (MWS-E) can transfer data to or receive data from the SSD-E. The SSD-E can only respond to transfer requests from these devices; the SSD-E cannot initiate a transfer.

The following subsections define the SSD-E memory, mainframe and SSD-E transfers, IOS-E and SSD-E transfers, and MWS-E and SSD-E transfers. A specification sheet is included at the end of this section.

SSD-E Memory

Each word contains 72 bits: 64 data bits and 8 check bits. The number of words varies, ranging from 128 Mwords to 2 Gwords. Table 4-1 lists the different memory sizes.

Table 4-1. SSD-E Memory Sizes

Model	Memory Size
SSD-128	128 Mwords
SSD-256	256 Mwords
SSD-512	512 Mwords
SSD-1024	1 Gword
SSD-2048	2 Gwords

To protect data, SSD-E memory uses single-error correction/double-error detection (SECDED †) logic.

When data is written into SSD-E memory, the SECDED logic generates a checkbyte (an 8-bit Hamming code) for each data word. The checkbyte and 64-bit data word are stored in the SSD-E memory.

When a word is read from SSD-E memory, a new checkbyte is generated for the data word. The new checkbyte is compared to the stored checkbyte. The result of the comparison is called the syndrome code. If the syndrome code equals 0, no data bits were altered, and the word is passed on to the external device.

If an error occurred, the SECDED logic analyzes the syndrome code to determine the number of altered data bits. If only a single data bit was altered, the SECDED logic toggles the bit to the correct state and passes the corrected word out to the external device. If two data bits were altered, the SECDED logic cannot correct the word, but it can detect the failure. If more than 2 bits are in error, the results are unpredictable. A message is sent to the error logger for all detected errors.

SSD-E and Mainframe Data Transfers

Data transfers between the SSD-E and the mainframe's central memory use very high-speed (VHISP) channels. Each VHISP channel simultaneously transfers two 64-bit words plus two 8-bit checkbytes. Each VHISP channel transfers data at the rate of 1,800 Mbytes/s.

The quantity of VHISP channels can range from one to four, depending on the quantity of mainframe CPUs. Typically, there is one VHISP channel for each CPU pair. The CRAY Y-MP C90 can support eight VHISP channels with 16 CPUs. Because an SSD-E can support 4 VHISP channels, a second SSD-E is required to use all eight VHISP channels from the mainframe.

To protect data, all VHISP channels use SECDED logic. This SECDED logic operates the same as the SECDED logic on the SSD-E memory.

Data transfers between the mainframe and the SSD-E are done in 64-word blocks. Individual words are not accessible by the mainframe. To read a particular word, an entire block is transferred and the word must be selected using software methods similar to disk storage data handling methods.

† Hamming, R. W. "Error Detection and Correcting Codes." *Bell System Technical Journal*. 29.2 (1950): 147–160.

All VHISP channels operate under mainframe program control. Programming a data transfer requires three parameters: the SSD-E starting address, the mainframe's central memory starting address, and a block length. The block length specifies how many 64-word blocks to transfer. The maximum block length is 16,777,216 which yields a maximum transfer length of 1,073,741,824 words.

SSD-E and IOS-E Data Transfers

Data transfers between the SSD-E and an I/O buffer in the IOS-E use high-speed (HISP) channel pairs. Each HISP channel pair consists of an input and output channel, both of which may be active simultaneously. Each channel is 64 bits wide and contains 8 check bits. Each channel transfers data at the rate of 200 Mbytes/s.

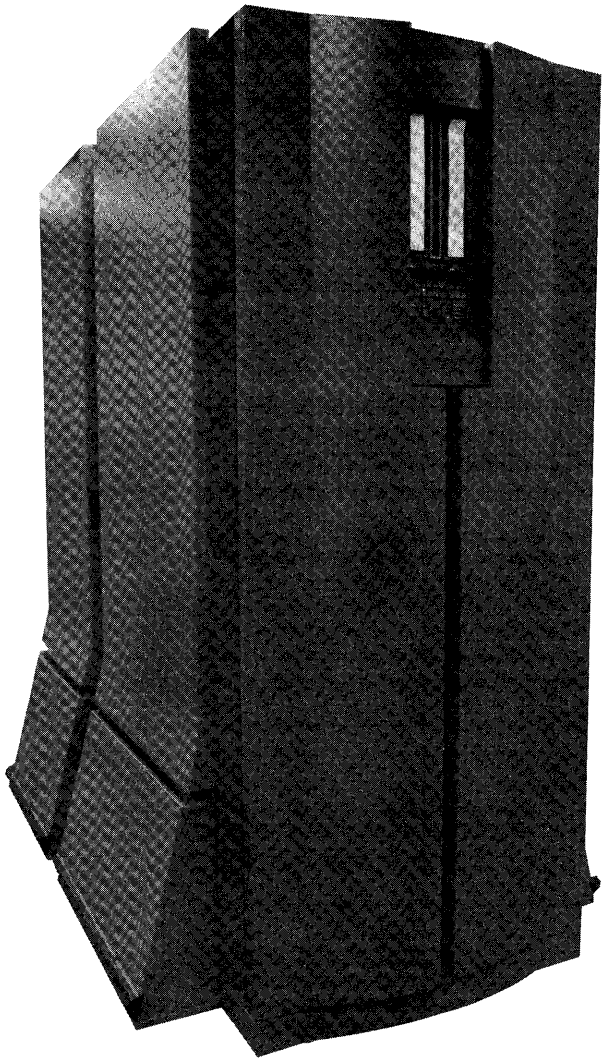
The quantity of HISP channel pairs ranges from two to eight. Typically, there is one HISP channel pair for each I/O cluster in the IOS-E.

To protect data, all HISP channels use SECDED logic. This SECDED logic operates the same as the SECDED logic on the SSD-E memory.

Data transfers between the IOS-E and the SSD-E are done in 64-word blocks. Individual words are not accessible by the IOS-E. To read a particular word, an entire block is transferred and the word must be selected using software methods similar to disk storage data handling methods.

All HISP channel pairs operate under IOS-E program control. Programming a data transfer requires three parameters: the SSD-E starting address, the selected I/O buffer's starting address, and a block length. The block length specifies how many 64-word blocks to transfer. The maximum block length is 1,024, which yields a maximum transfer length of 65,536 words.

SSD Model E Specifications



CRAY RESEARCH, INC.

SSD-E Specifications

General Specifications

Storage word size . . . . . 72 bits

Data block size . . . . . 64 decimal words per transfer

Maximum block operation:
- 16,777,215 decimal blocks (77777777_8) for VHISP channels
- 256 decimal blocks (377_8) for HISP channels

Storage capacity:
- 128 Mwords – 1 Mbit DRAM
- 256 Mwords – 1 Mbit DRAM
- 512 Mwords – 1 Mbit DRAM
- 1 Gword – 4 Mbit DRAM
- 2 Gword – 4 Mbit DRAM

Memory configurations:
- 128 Mword - 1 section, 4 groups, 1 Mbit DRAMs
- 256 Mword - 2 section, 8 groups, 1 Mbit DRAMs
- 512 Mword - 4 section, 16 groups, 1 Mbit DRAMs
- 1 Gword - 2 section, 8 groups, 4 Mbit DRAMs
- 2 Gword - 4 section, 16 groups, 4 Mbit DRAMs

Maximum band width:
- 100 Gbits/s reading and writing one word per group/clock period

User ports:
- 4 VHISP channel pairs 1,800 Mbytes/s each
- 8 HISP channel pairs 200 Mbytes/s each

Data protection:
- Single-error correction/double-error detection (SECDED) before and after storage and on all user channels.

Physical Description an SSD-E Chassis

Height . . . . . 76.25 in. (194 cm)
Width . . . . . 46 in. (117 cm)
Depth . . . . . 73.5 in. (187 cm)
Weight . . . . . 7,695 lbs (3,182 kgs)
Floor loading . . 520 lbs/ft^2 (2,538 kg/m^2)
Access requirements . . . . . 3 ft (0.9 m) on all sides
Heat dissipation to air . . . . . 3.15 kW (maximum)

5 PERIPHERAL EQUIPMENT

The following subsections describe the major components of the disk drives and various network interfaces used with the CRAY Y-MP C90 computer system.

Disk Controller Units and Disk Storage Units

Disk systems provide long-term data storage for a Cray Research computer system. Components of the disk system include disk channel adapters and disk drives. DCA-1 disk channel adapters control one or two DD-40 or DD-41 disk drives. DCA-2 disk channel adapters control from one to eight DD-60 or DD-61 disk drives. Refer to Section 3 of this manual for more information on channel adapters.

Disk Drives

The disk drives store data on magnetic disks. Typically, a disk drive consists of several rotating platters. Data is accessed by read and write heads organized into groups. Heads are controlled and positioned by one or more head actuator (servo) mechanisms on the disk cylinders. The following subsections describe specific disk drives.

DD-60 Disk Drive

One DD-60 disk drive consists of a single-sealed head disk assembly (HDA). The HDA contains 2 sets of 9 parallel read and write heads, 1 servo head, 11 eight-inch rotating platters, and 20 thin film media surfaces.

One set of nine parallel heads is used at a time for data transfers. Eight of the heads are used for data and the ninth head is used for parity. The heads can be positioned over 2,608 user-accessible locations on the surface of each platter. Each location is called a cylinder. The DD-60 determines which cylinder the heads are on by reading the information under the servo head.

When the heads are stationary, the area under one head during one complete revolution of the platter is called a track. The DCA-2 disk channel adapter combines eight tracks of data (one from each head) into one logical track. A logical track contains 23 sectors where data can be stored and from which data can be retrieved by the operating system.

Data in one sector is called a data block. One data block consists of 2,048 64-bit words of IOP data plus verification and error-correcting data. Data is transferred between the disk surface and I/O buffer in the IOP in blocks of this fixed size.

One DE-60 disk enclosure cabinet contains a maximum of nine DD-60 disk drives. Eight of the disk drives store system data, and the ninth disk drive is a spare. The DCA-2 disk channel adapter in the IOP manages control signals and protocol for the individual disk drives in a DE-60.

The DCA-2 performs the following functions:

- Controls up to eight DD-60 disk drives (daisy chain configuration)
- Passes control functions to the selected drives
- Receives status from the drives
- Generates codes for correcting write data errors
- Checks read data correcting codes and corrects read data when necessary

Initially, a factory flaw table is used to locate media flaws on the surface of a platter. If additional flaws are found, diagnostic programs determine the location and width of the flaw. These flaws, which are identified during surface analysis, are avoided during read and write operations. A defect parameter in the sector ID field contains information on the location of the flaw.

Under control of a DCA-2, a DD-60 writes data into a flawed sector until the media defect location is reached. While the read and write head of the DD-60 is over the media defect, it writes a copy of the previous 18 bytes of data. Then the DD-60 resumes writing valid data in the flawed sector.

When reading data from a flawed sector, a DD-60 reads the defect address to find the beginning of the 18-byte field of repeated data. When the read and write head of the DD-60 reaches the field of repeated data, the DCA-2 does not accept the repeated data. The drive resumes its normal read operation after the head passes the defect field.

DD-60 Single-port Configuration

A single-port configuration connects one DCA-2 to one disk drive. In this configuration, the channel accesses information at the maximum data transfer rate of the disk drive. Because only one disk drive connects to the channel, the storage capacity of the channel is the storage capacity of the disk drive.

Figure 5-1 shows eight disk drives, each connected in a single-port configuration. One DCA-2 connects to the input of port A, and a terminator connects to the output of port A for each disk drive. Port B is not used.

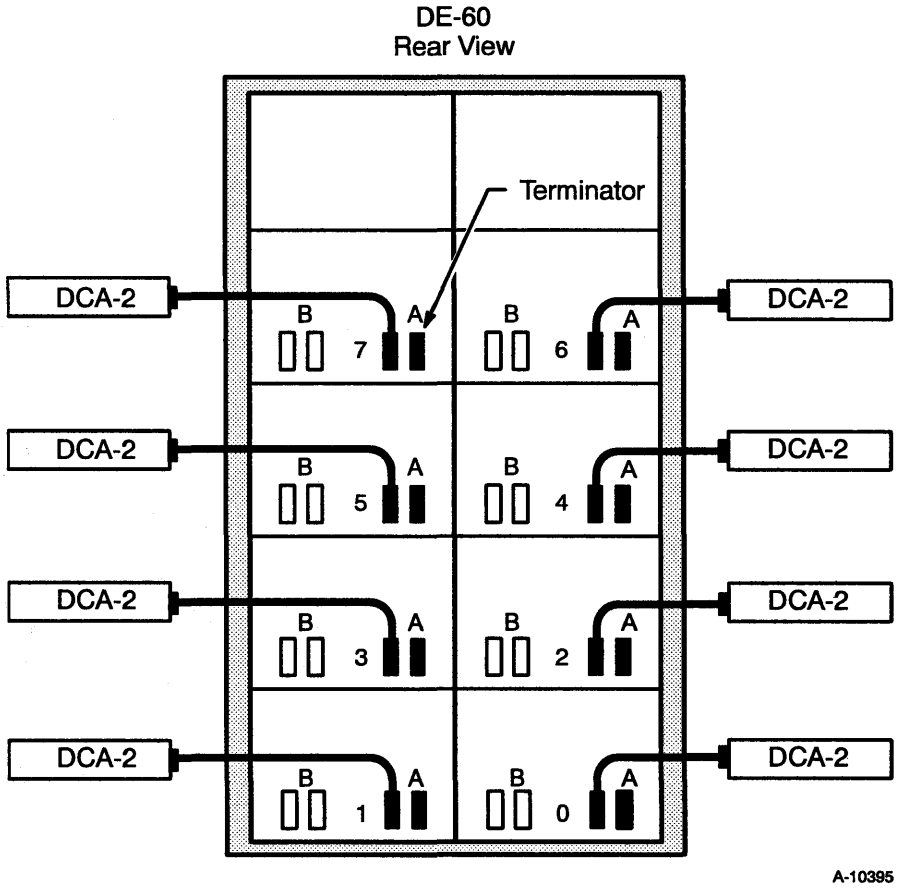


Figure 5-1. DD-60 Single-port Configurations

DD-60 Daisy Chain Configuration

A DD-60 daisy chain configuration (refer to Figure 5-2) consists of a maximum of eight DD-60 disk drives connected to one channel. The storage capacity per channel is multiplied by the number of drives connected in the daisy chain; however, only one DD-60 can transfer data to the DCA-2 at a time.

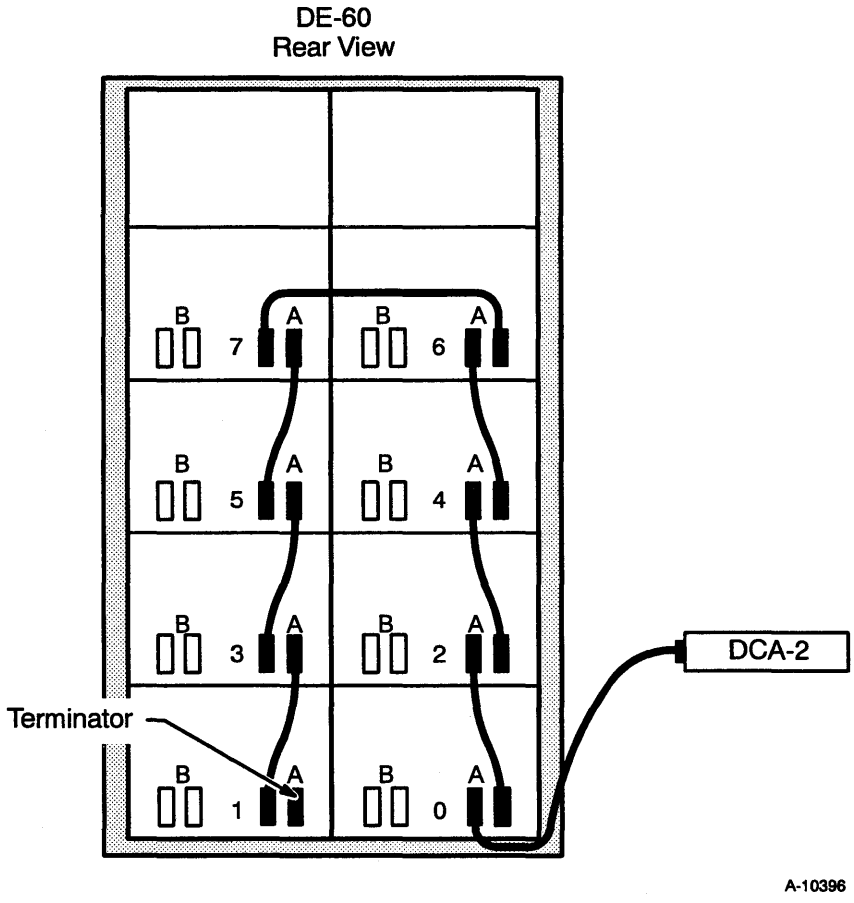


Figure 5-2. DD-60 Daisy Chain Configuration

DD-60 Alternate-path Configuration

An alternate-path configuration connects two DCA-2s to a maximum of eight disk drives. In this configuration, the two DCA-2s connect to the same set of disk drives on two separate daisy chains. Special software modifications must be made when the disk drives are cabled in a redundant configuration.

Figure 5-3 shows eight disk drives connected in redundant configurations. Each disk drive has one DCA-2 connected to port A (primary path) and another DCA-2 connected to port B (redundant path).

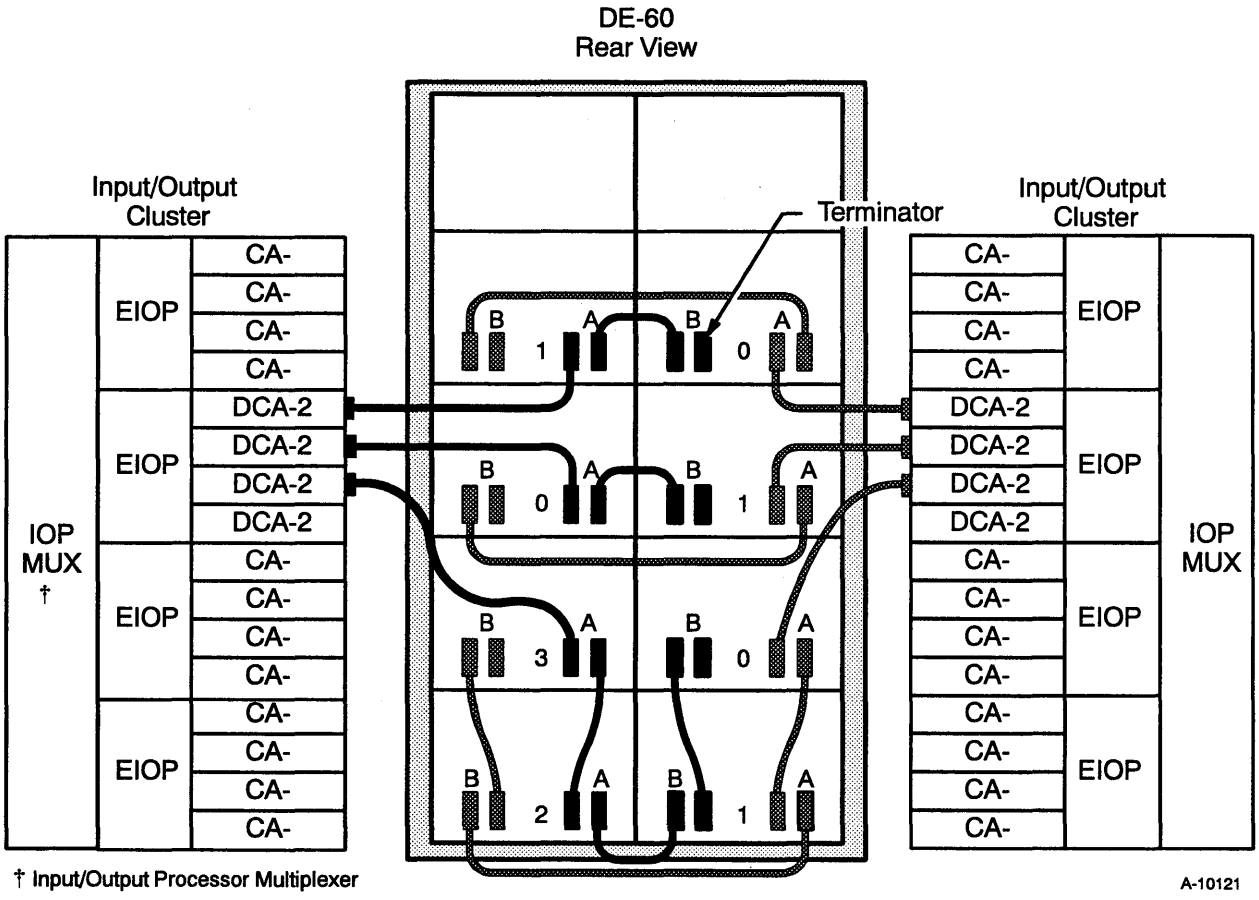


Figure 5-3. DD-60 Alternate-path Configurations

DD-61 Disk Drive

The DD-61 disk drive is a 19-head serial disk drive similar to the DD-60. During data transfers to and from the DCA-2, one head transfers data at a time. The DD-61 has a sustained transfer rate of 2.6 Mbytes/s and a storage capacity of 2.23 Gbytes.

One DD-61 disk drive includes a sealed HDA. The HDA contains 19 serial read and write heads, 1 servo head, 11 eight-inch rotating platters, and 20 thin film media surfaces.

The heads can be positioned over 2,608 user-accessible locations on the surface of each platter. Each location is called a cylinder. The DD-61 determines which cylinder the heads are on by reading the information under the servo head.

When the heads are stationary, the area under one head after one complete revolution of the platter is called a track. A track contains 11 sectors where data can be stored and from which data can be retrieved by the operating system.

Data in one sector is called a data block. One data block consists of 512 64-bit words of IOP data plus verification and error-correcting data. Data is transferred between the disk surface and the I/O buffer in the IOP in blocks of this fixed size. Sectors may be chained during both read and write operations.

One DE-60 disk enclosure cabinet contains a maximum of eight DD-61 disk drives. The DCA-2 disk channel adapter in the IOP manages control signals and protocol for the individual disk drives in a DE-60.

The DCA-2 performs the following functions:

- Controls a maximum of eight DD-61 disk drives (daisy chain configuration)
- Passes control functions to the selected drives
- Receives status from the drives
- Generates codes for correcting write data errors
- Checks read data correcting codes and corrects read data when necessary

Initially, a factory flaw table is used to locate media flaws on the surface of a platter. If additional flaws are found, diagnostic programs determine the location and width of the flaw. These flaws, which are identified during surface analysis, are avoided during read and write operations. A defect parameter in the sector ID field contains information on the location of the flaw.

Under control of a DCA-2, a DD-61 writes data into a flawed sector until the media defect location is reached. While the read and write head of the DD-61 is over the media defect, it writes a copy of the previous 18 bytes of data. Then the DD-61 resumes writing valid data in the flawed sector.

When reading data from a flawed sector, a DD-61 reads the defect address to find the beginning of the 18-byte field of repeated data. When the read and write head of the DD-61 reaches the field of repeated data, the DCA-2 does not accept the repeated data. The drive resumes its normal read operation after the head passes the defect field.

DD-61 disk drives also connect in the same single-port, daisy chain, or alternate-path configurations as DD-60s. For information on the daisy chain and alternate-path configurations, refer to the "DD-60 Disk Drive" subsection in this section.

DS-41 Disk Subsystem

The DS-41 disk subsystem consists of the the DC-41 disk controller and the DD-41 disk drive. Each DD-41 disk drive has four spindles that operate as a single logical disk drive unit under the control of one DC-41 disk controller. Each physical disk drive (spindle) consists of 9 rotating platters and 15 recording surfaces. Data is accessed by 15 read and write heads. A servo mechanism, which controls the read and write heads, positions the heads over one of 1,635 disk cylinders. Data is stored and retrieved from the recording surface of the disk drive by any of the 15 read and write heads.

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 stored and retrieved by the operating system. The data in one sector is called a data block. One data block consists of 2,048 16-bit parcels (512 64-bit words) of IOP data plus verification and error-correcting data. Data can be transferred between the disk surface and local memory in the IOP only in blocks of this fixed size. Sectors may be chained for both read and write operations.

A DC-41 disk controller provides interface logic to adapt DCA-1 signals and protocol for individual disk drive units, to handle routing among the drives, and to buffer data from the four spindles in a full-track buffer. The interface logic in one DC-41 disk control unit performs the following functions:

- Controls up to 8 spindles (two DD-41 disk drives)
- Passes control functions to the selected drives
- Passes status from the drives to the DCA-1
- Buffers read and write data for transfers between DCA-1 and the disk drives
- Generates error-correcting codes for write data
- Checks read data correcting codes and corrects read data when necessary
- Controls distribution of read and write data over 48 sectors per track using 12 sectors from each of the four spindles in a logical drive

Under the control of a DC-41, a disk drive writes data to a flawed sector until a defect location is reached. In the area starting at a defect address, a disk drive writes a 16-byte field of 0's. A disk drive resumes writing data in a flawed sector following this field of 0's. When reading data from a flawed sector, a disk drive reads the defect address to find where the field of 16 bytes of 0's starts. When a drive's read and write head reaches the field of 0's, the head skips over the flawed area of the sector overwritten by the field of 0's, omitting them from the read data. The drive resumes its normal read operation after the head passes the defect field.

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. Defective areas of the recording surface, which are identified during surface analysis, are avoided during read and write operations. A defect parameter becomes part of the sector ID field when the drive is formatting.

DS-41 Single-port Configuration

A single-port configuration connects one DC-41 to one DD-41. In this configuration, the channel accesses information at the maximum data-transfer rate of the DD-41. Because only one disk drive connects to the channel, the storage capacity of the channel is the storage capacity of the disk drive.

DS-41 Daisy Chain Configuration

A daisy chain configuration connects one DC-41 to two DD-41s. The channel data-storage capacity is the total storage capacity of both disk drives. Because only one disk drive can transfer data to the DC-41 at a time, the channel data transfer rate is the maximum transfer rate of one disk drive.

DS-41 Alternate-path Configuration

An alternate-path configuration connects two DC-41s to a maximum of two disk drives. In this configuration, the two DC-41s connect to the same disk drives on two separate daisy chains. Special software modifications must be made when the disk drives are cabled in an alternate-path configuration.

DS-41A Disk Subsystem Field-upgradable Configurations

A field-upgradable DS-41A disk subsystem configuration has the following components:

- One disk drive cabinet (DE-41)
- One DD-41 disk drive (housed in the DE-41)
- One disk controller cabinet (DCC-2A)
- One DC-40 disk controller (housed in the DCC-2A)

A DS-41A can be upgraded by adding a DS-41B package. A DS-41B consists of one DD-41, one DC-41, and all cabling required to install the additional drive and controller in a DS-41A disk subsystem. Up to three DS-41Bs can be installed in a DS-41A disk subsystem.

DS-40 Disk Subsystem

The DS-40 disk subsystem comprises the following components: the DD-40 disk drive, the DC-40 disk control unit (DCU), and the disk controller cabinet (DCC-2). The DD-40 contains four disk drives and 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 "DS-40 and DS-40D Disk Subsystem Specifications" at the end of this section for exact configuration information. Each physical disk drive (spindle) consists of six rotating platters and ten recording surfaces. Data is accessed by 19 read and write heads that are 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. The data in one sector is called a data block. One data block consists of 2,048 16-bit data parcels (512 64-bit words) plus verification and error-correcting data. Data can be transferred between the disk surface and I/O buffer in the IOP only in blocks of this fixed size. Sectors may be chained for both read and write operations.

Interface logic in the DC-40 also adapts the DCA-1 signals and protocol to the individual disk drive units, manages routing among the drives, and buffers the data from the four drives in a full-track buffer.

The interface logic in one DC-40 disk control unit performs the following functions:

- Controls up to 8 spindles (two DD-40 disk storage units)
- Passes control functions to the selected drives
- Passes status from the drives to the DCA-1
- Buffers read and write data for transfers between DCA-1 and the disk drives
- Generates error-correcting codes for write data
- Checks read data correcting codes and corrects read data when necessary
- Controls distribution of read and write data over 48 sectors per track using 12 sectors from each of the four spindles in a logical drive

Under the control of a DC-40, a disk drive writes data onto a flawed sector until a defect location is reached. In the area starting at a defect address, a disk drive writes a 16-byte field of 0's. A disk drive resumes writing data in a flawed sector following this field of 0's. When reading data from a flawed sector, a disk drive reads the defect address to find where the field of 16 bytes of 0's starts. When a drive's read and write head reaches the field of 0's, the head ignores the field of 0's, omitting the field of 0's from the read data. The drive resumes its normal read operation after the head passes the defect field.

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. Defective areas of the recording surface, which are identified during surface analysis, are avoided during read and write operations. A defect parameter becomes part of the sector ID field when the drive is formatting.

DS-40 Single-port Configuration

A single-port configuration connects one DC-40 to one DD-40. In this configuration, the channel accesses information at the maximum data-transfer rate of the DD-40. Because only one disk drive connects to the channel, the storage capacity of the channel is the storage capacity of the disk drive.

DS-40 Daisy Chain Configurations

A daisy chain configuration connects one DC-40 to two DD-40s. The channel data-storage capacity is the total storage capacity of both disk drives. Because only one disk drive can transfer data to the DC-40 at a time, the channel data-transfer rate is the maximum transfer rate of one disk drive.

DD-49 Disk Drive

The DD-49 disk drive consists of nine rotating platters. Data is accessed by 32 read and write heads organized into eight groups, four read and 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, and 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 (512 64-bit words) of IOP data plus verification and error1-correction data. Data can be transferred between the disk surface and I/O buffer in the IOP only in blocks of this fixed size. Sectors may be chained for both read and write operations.

The DD-49 disk drive responds to commands from the IOS-E 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 disk drive 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 sectors n through 41 must be re-created. 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 offline. A hardware diagnostic utility reformats the track that contains slipped sectors.

A DD-49 disk drive has 44 sectors per track, although only 42 sectors are used for data. Under normal circumstances, the two spare sectors are not used. If one of the data sectors becomes flawed, however, a spare sector is used as a data sector.

Refer to "DD-49 Disk Drive Specifications" at the end of this section for configuration information.

Network Interfaces

The CRAY Y-MP C90 computer system can be connected to a wide variety of computer systems (often referred to as "front-end systems") and networks through a CCA-1 channel adapter in the IOS. This enables users of non-Cray Research computer systems to utilize the CRAY Y-MP C90 system's extraordinary computational power. The following subsections describe the methods used to interface the CRAY Y-MP C90 computer system with other computer systems and networks.

FEI-1 Front-end Interface

The FEI-1 front-end interface provides communication between a CCA-1 channel adapter in the IOS 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 "Front-end Interface Specifications" 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 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 Research programmer.

Fiber-optic Link

The Cray Research fiber-optic link (FOL) is used as a channel extender for 6-Mbyte/s (16-bit asynchronous) channels. It connects the conventional wire cable from the CCA-1 channel adapter in the IOS to the wire cable from an FEI-1. 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 4,000 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 Research 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 smaller cabinet containing more fiber-optic modules is located next to the Cray Research mainframe. These special fiber-optic modules modulate and demodulate the signals between the Cray Research mainframe and the front-end system.

FEI-3 Front-end Interface

The FEI-3 is a group of similar front-end interfaces that enables VME-based microcomputers and workstations to communicate with a CCA-1 channel adapter in the IOS over a standard 6-Mbyte/s I/O channel. Specific FEI-3 applications depend on the capabilities of the VME workstations or microcomputers. For example, Cray Research uses the FEI-3 to connect systems to an operator workstation.

The following list contains other possible FEI-3 applications:

- To connect to a communications gateway for Control Subsystem Networks or other networks
- To connect to a graphics output processor or device
- To connect to a remote Cray Research 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 customer furnishes and provides support for the target VME system.

The VMEbus is an industry standard that specifies the electrical and mechanical rules for a microcomputer backplane. Many popular microcomputer systems are based on the VMEbus.

Direct Network Connections

The CCA-1 channel adapter in the IOS supports direct connection to network adapters such as Network Systems Corporation (NSC) HYPERchannel adapters, Computer Network Technology LANlord adapters, and others.

High Performance Parallel Interface (HIPPI)

The Cray Research High Performance Parallel Interface (HIPPI) is an external channel that provides high-speed communications between HCA-3 and HCA-4 channel adapters in the IOS and peripheral equipment, such as network adapters, raster display devices, and mass storage systems. HIPPI conforms to industry standards and provides 32-bit parallel data transfers at the rate of 100-Mbytes/s.

HIPPI conforms to the preliminary draft proposed American National Standard (DPANS) HIPPI revision 7.0. The HIPPI proposal is based on an original design by engineers at Los Alamos National Laboratories.

HIPPI is a simplex channel that transmits data in one direction; it is usually configured in pairs for full duplex operation. Driver software enables users to operate the HIPPI directly as a raw device or indirectly through Transmission Control Protocol/Internet Protocol (TCP/IP) or User Datagram Protocol (UDP) sockets, Remote Procedure Call (RPC) libraries, and Network File Systems (NFSs) between Cray Research computer systems.

Because HIPPI conforms to industry standards, it can be configured with many types of devices and applications that require high-speed transfer of large amounts of data.

The following list contains other HIPPI applications:

- Distributed applications. The speed of HIPPI makes more applications suitable for distributed processing. Users can link multiple Cray Research computer systems for maximum supercomputer performance.

- Raster graphics. Real-time animated graphics are possible when HIPPI is combined with a compatible high-speed frame buffer. Existing devices have delivered up to 60 frames per second on a 512-by-512 raster of 24-bit pixels.

DEC VAX Supercomputer Gateway

Digital Equipment Corporation (DEC) offers a VAX Supercomputer Gateway to enable direct connection between the DEC VAXcluster environment and a CCA-1 channel adapter in the IOS.

DD-60 Disk Drive Specifications

CRAY RESEARCH, INC.

DD-60 Specifications

DD-60 Features

Transfer rate:
- Sustained ..... 16 to 20 Mbytes/s
- Burst rate ..... 24 Mbytes/s

Storage capacity
- One DD-60 ..... 1.96 Gbytes

Total data sectors 119,968
Logical sector size (64-bit words) ... 2,048

Total data words 245,694,464

Typical position delays
- Single track ..... 2.2 ms
- Average ..... 12 ms
- Full stroke ..... 26 ms

Average latency ..... 8.3 ms

DE-60 Power and Cooling Specifications (eight DD60s)

Required power ..... 3 phase, 208 Vac, 50 or 60 Hz, 20 A per phase or 3phase, 400 Vac, 50 or 60 Hz, 8.5 A per phase

Heat load ..... 7,500 Btu/hr (2,200 W)

Type of cooling ..... air cooled

DE-60 Physical Description

Dimensions
- Height ..... 61.7 in. (156.7 cm)
- Width ..... 24 in. (61 cm)
- Depth ..... 39.2 in. (99.5 cm)

Floor space ..... 6.5 ft^2 (0.6 m^2)

Weight
- (all eight DD-60s) .... 1,015 lbs (460 kg)

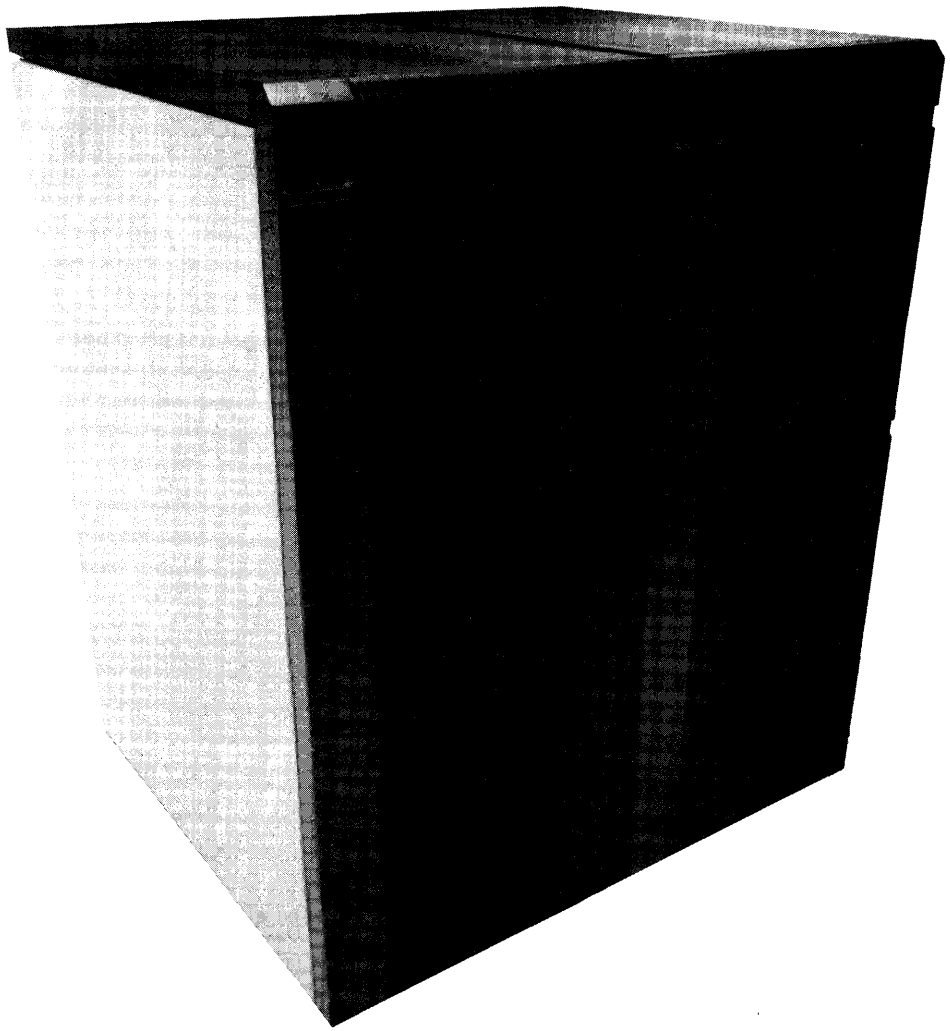
DE-60 Placement and Cabling Specifications

Minimum clearance
- Sides ..... 12 in. (25 cm)
- Front ..... 36 in. (91 cm)
- Back ..... 36 in. (91 cm)

Length of power cable ..... 6 ft (1.8 m)

Maximum length of
- data cables ..... 82 ft (25 m)

DD-61 Disk Drive Specifications



CRAY RESEARCH, INC.

DD-61 Specifications

DD-61 Features

Transfer rate:
- Sustained ..... 2.3 to 2.6 Mbytes/s
- Burst rate ..... 3.0 Mbytes/s

Storage capacity
- One DD-61 ..... 2.23 Gbytes

Total data sectors 545,072

Logical sector size (64-bit words) 512

Total data words 279,076,864

Typical position delays:
- Single track ..... 2.2 ms
- Average ..... 12 ms
- Full stroke ..... 26 ms

Average latency ..... 8.3 ms

DE-60 Power and Cooling Specifications (eight DD60s)

Required power ..... 3 phase, 208 Vac, 50 or 60 Hz, 20 A per phase or 3 phase, 400 Vac, 50 or 60 Hz, 8.5 A per phase

Heat load ..... 3,900 Btu/hr (1,144 W)

Type of cooling ..... air cooled

DE-60 Physical Description

Dimensions
- Height ..... 61.7 in. (156.7 cm)
- Width ..... 24 in. (61 cm)
- Depth ..... 39.2 in. (99.5 cm)

Floor space ..... 6.5 ft^2 (0.6 m^2)

Weight
- (all eight DD-61s) ..... 900 lbs (408 kg)

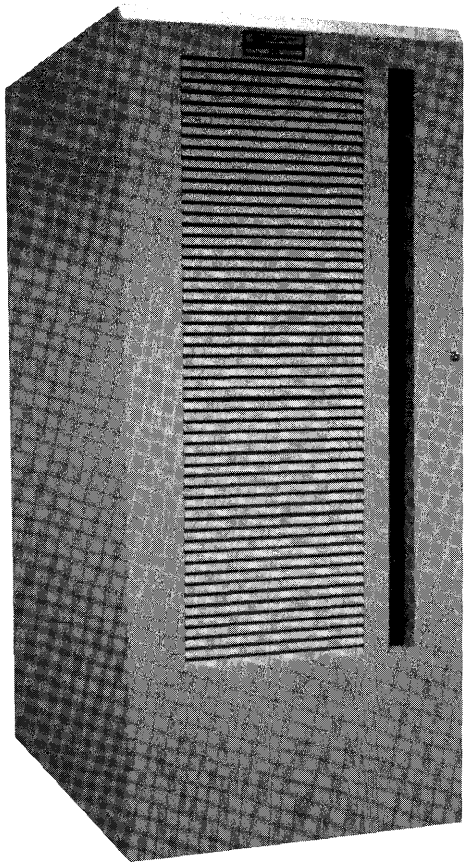
DE-60 Placement and Cabling Specifications

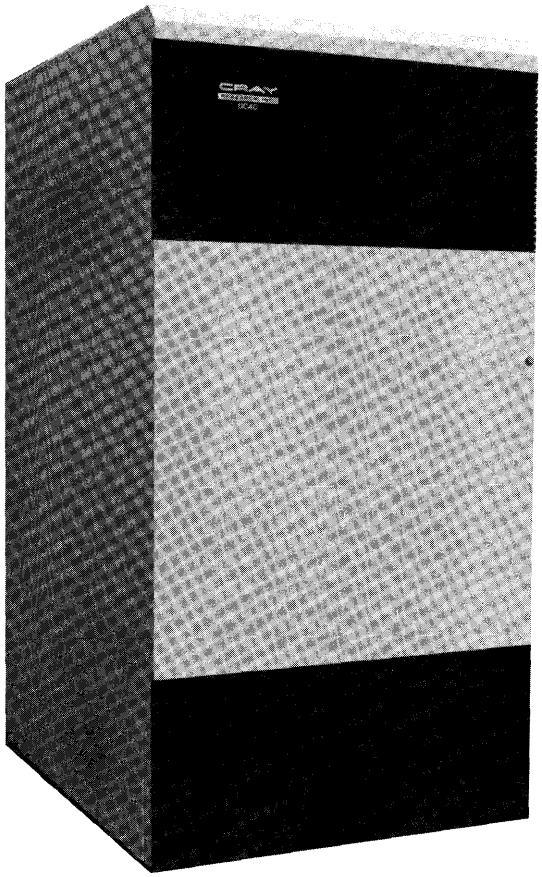
Minimum clearance
- Sides ..... 12 in. (25 cm)
- Front ..... 36 in. (91 cm)
- Back ..... 36 in. (91 cm)

Length of power cable ..... 6 ft (1.8 m)

Maximum length of
- data cables ..... 82 ft (25 m)

DS-40 and DS-40D Disk Subsystem Specifications





CRAY RESEARCH, INC.

DS-40 and DS-40D Specifications

DC-40 Features

Transfer rate:
- Sustained ..... 9.6 Mbytes/s
- Burst rate ..... 20 Mbytes/s

Storage capacity ..... 5,200 Mbytes

DC-40 Power and Cooling

Required power ..... 208 Vac, 3 phase, 60 Hz, 60 A
Type of cooling ..... water cooled refrigeration/air cooling
Water temperature (°F) ..... 40 to 90
Water temperature (°C) ..... 4.4 to 32.2
Heat load (to air) ..... 1,330 Btu/hr, 390 W
Heat rejection to water ..... 24,000 Btu/hr, 7,643 W

DCC-2/DC-40 Physical Description

The four 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 ..... 8.7 ft^2 (0.81 m^2)
Weight ..... 1,240 lbs (562 kg)
Cabinet dimensions:
- Height ..... 60 in. (152 cm)
- Width ..... 31 in. (79 cm)
- Depth ..... 41 in. (104 cm)

DCC-2/DC-40 Placement and Cabling

Minimum clearance:
- Sides ..... 12 in. (30.5 cm)
- Front ..... 36 in. (91.4 cm)
- Back ..... 36 in. (91.4 cm)

Length of power cable ..... 8 ft (2.4 m)
Maximum length of data cables ..... 50 ft (14.4 m)

DD-40 Features

Transfer rate:
- Sustained ..... 9.6 Mbytes/s
- Burst rate ..... 20 Mbytes/s

DD-40 Features (continued)

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

DD-40 Power and Cooling

Required power ..... 208 Vac, 3 phase, 60 Hz, 20 A
Cooling ..... air cooled
Heat load ..... 8,000 Btu/hr, 2,340 W

DD-40 Physical Description

Floor space ..... 7.3 ft^2 (0.68 m^2)
Weight ..... 1,150 lbs (522 kg)
Cabinet dimensions:
- Height ..... 60 in. (152 cm)
- Width ..... 26 in. (66 cm)
- Depth ..... 41 in. (104 cm)

DD-40 Placement and Cabling

Minimum clearance:
- Sides ..... 1 in. (2.5 cm)
- Front ..... 36 in. (91.4 cm)
- Back ..... 30 in. (76.2 cm)

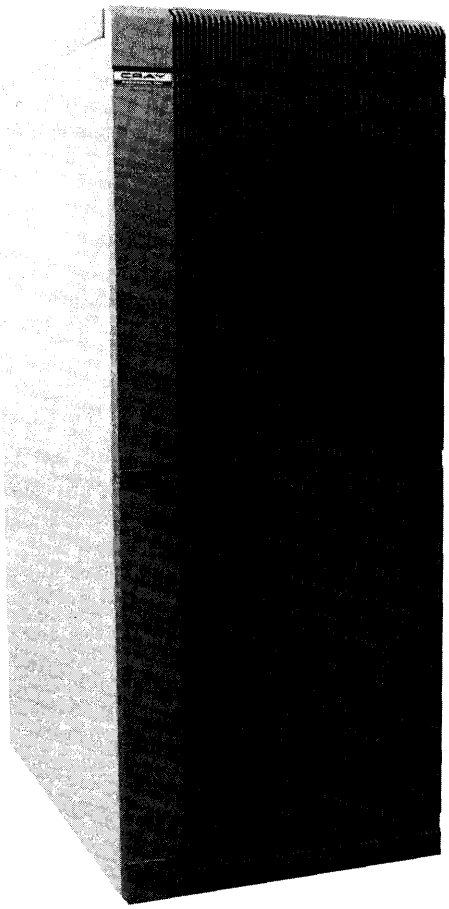
Length of power cable ..... 6 ft (1.8 m)
Maximum length of data cables ... 20 ft (6 m)

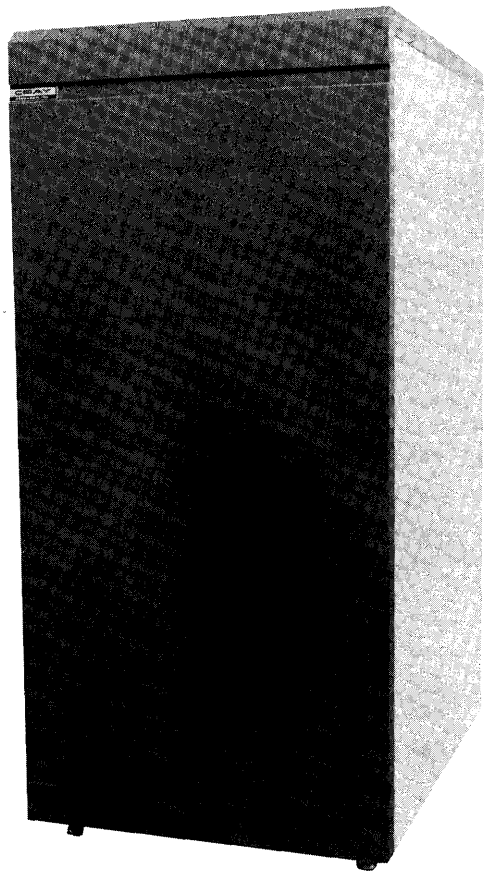
The DCC-2 contains four DC-40 disk controllers. The DC-40 is a dual-ported interface with only one port active at a time.

Four disk storage units (DSUs) are connected to the DCC-2 chassis for the DS-40 Disk Subsystem.

Eight DSUs are connected to the DCC-2 chassis for the DS-40D disk subsystem, but only four DSUs can be active at one time. This technique, known as daisy chaining, is used to double the capacity of a single subsystem from 21 Gbytes to 42 Gbytes; doubling the capacity does not double the performance because the data path is set at 9.6 Mbytes/s.

DS-41, DS-41D, and DS-41R Disk Subsystem Specifications





CRAY RESEARCH, INC.

DS-41, DS-41D, and DS-41R Specifications

DC-41 Features

Sustained transfer rate ..... 9.6 Mbytes/s
Storage capacity ..... 4,800 Mbytes

DCC-2A Power and Cooling
(four DC-41s)

Required power ..... 208 Vac, 3 phase, 60 Hz, 60 A
Type of cooling ..... water cooled refrigeration/air cooling
Water temperature (°F) ..... 40 to 90
Water temperature (°C) ..... 4.4 to 32.2
Heat load (to air) ..... 1,330 Btu/hr, 390 W
Heat rejection
to water ..... 24,000 Btu/hr, 7,643 W

DCC-2A Physical Description
(four DC-41s)

The DC-41s are housed in a disk control cabinet (DCC-2A) that contains the power control and refrigeration components required for the DC-41.

Floor space ..... 8.7 ft^2 (0.81 m^2)
Weight ..... 1,240 lbs (562 kg)
Cabinet dimensions:
Height ..... 67 in. (170 cm)
Width ..... 31 in. (79 cm)
Depth ..... 41 in. (104 cm)

DCC-2A Placement and Cabling

Minimum clearance:
Sides ..... 12 in. (30.5 cm)
Front ..... 36 in. (91.4 cm)
Back ..... 36 in. (91.4 cm)
Length of power cable ..... 8 ft (2.4 m)
Maximum length of
data cables ..... 50 ft (14.4 m)

The DCC-2A contains four DC-41 controllers. The DC-41 is a dual-ported interface with only one port active at a time. Two DCC-2A cabinets are used in a DS-41R disk subsystem to provide dual-channel access.

Storage capacity and transfer rates are the same for both DS-41D and DS-41R disk subsystems.

DD-41 Features

Sustained transfer rate ..... 9.6 Mbytes/s

Total data sectors 1,175,760
Total data words 601,989,120
Typical position delays:
Single track ..... 5 ms
Average ..... 16 ms
Full stroke ..... 30 ms

DE-41 Power and Cooling
(four DD-41s)

Required power ..... 208 Vac, 3 phase, 60 Hz, 20 A
Cooling ..... air cooled
Heat load ..... 8,000 Btu/hr, 2,340 W

DE-41 Physical Description
(four DD-41s)

Floor space ..... 7.3 ft^2 (0.68 m^2)
Weight ..... 1,150 lbs (522 kg)
Cabinet dimensions:
Height ..... 67 in. (170 cm)
Width ..... 26 in. (66 cm)
Depth ..... 41 in. (104 cm)

DD-41 Placement and Cabling

Minimum clearance:
Sides ..... 1 in. (2.5 cm)
Front ..... 36 in. (91.4 cm)
Back ..... 30 in. (76.2 cm)

Length of power cable ..... 6 ft (1.8 m)
Maximum length of data cables ... 20 ft (6 m)

Four disk drives are connected to the DCC-2A chassis for the DS-41 disk subsystem. Eight disk drives are connected to the DCC-2A chassis for the DS-41D disk subsystem, but only four disk drives can be active at one time. This technique, known as daisy chaining, is used to double the capacity of a single subsystem from 19.2 Gbytes to 38.4 Gbytes; daisy chaining does not increase the 9.6-Mbyte/s transfer rate.

DD-49 Disk Drive Specifications

CRAY RESEARCH, INC.

DD-49 Specifications

DD-49 Features

Storage capacity ..... 1,200 Mbytes

Transfer rate:
- Sustained ..... 9.6 Mbytes/s
- Burst rate ..... 12 Mbytes/s

Total data sectors 297,696
Total data words 150,420,352

Typical position delays:
- Single track ..... 2 ms
- Average ..... 16 ms
- Full stroke ..... 30 ms

Power and Cooling

Required power ..... 3 phase, 208 Vac, 60 Hz, 20 A per phase
Heat load ..... 9,000 Btu/hr (2,640 W)
Type of cooling ..... air cooled

Physical Description

Floor space ..... 7.3 ft^2 (0.68 m^2)
Weight ..... 844 lbs (383 kg)

Placement and Cabling

Minimum clearance
- Sides ..... 12 in. (25 cm)
- Front ..... 36 in. (91 cm)
- Back ..... 30 in. (76 cm)

Length of power cable ..... 6 ft (1.8 m)
Maximum length of data cables ..... 50 ft (15 m)

Front-end Interface Specifications

CRAY RESEARCH, INC.

FEI Specifications

FEI Features

Cray Research, Inc. offers hardware interfaces and station software to connect the CRAY Y-MP C90 system 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
- Network Systems Corporation
  - HYPERchannel

Workstations:
- Sun-3 (through FEI-3's interface)

Operating systems:
- Apollo AEGIS
- CDC NOS, NOS/BE, and NOS/VE
- Data General AOS
- Data General RDOS
- DEC VAX/VMS
- Bull HN Information Systems
- IBM MVS and VM
- Unisys
- UNIX

Physical Description

Floor space . . . . . 4.38 ft^2 (3.42 m^2)
Weight . . . . . 200 lbs (91 kg)
Height . . . . . 23 in. (58.4 cm)

FOL-3 Fiber-optic Link Specifications

CRAY RESEARCH, INC.

FOL-3 Specifications

FOL-3 Description

The FOL-3 is a fiber-optic connection between a Cray Research I/O subsystem (IOS) and a front-end interface (FEI). The FOL-3 is an alternative to the wire cabling between an IOS and an FEI. The FOL-3 is designed primarily to increase the maximum cabling distance between a Cray Research computer system and a front-end computer and to provide complete electrical isolation from electromagnetic fields.

FOL-3 Configurations

The FOL-3 consists of the following equipment:

- Fiber-optic (FO) cabinet
- Interface (IO) cabinet
- Electrical kit

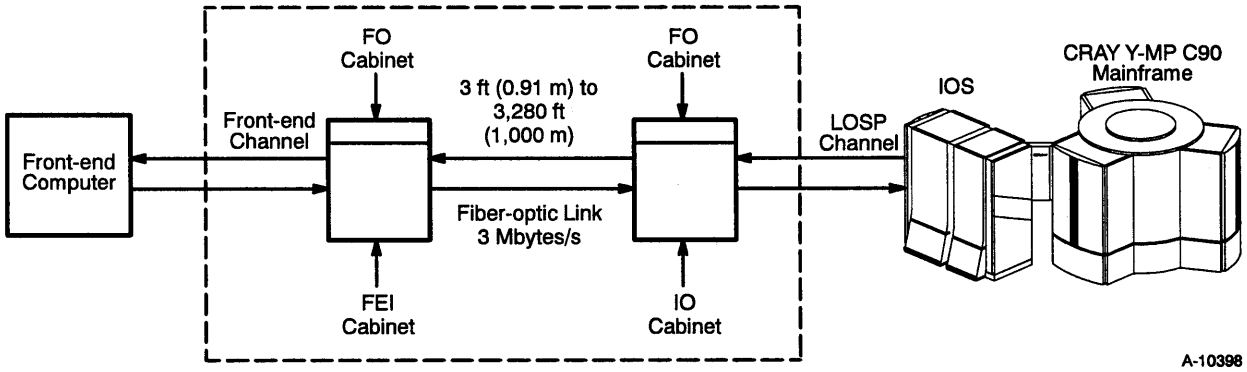
Below is an illustration of a general configuration of the FOL-3 used with an IOS. The dashed line encircles the components that comprise the FOL-3.

At the Cray Research mainframe end (local end of the FOL) is a fiber-optic cabinet that consists of an IO cabinet and an FO cabinet. The FO cabinet is positioned on top of the IO cabinet. The FO cabinet contains an FO module that includes the receivers and transmitters for the fiber-optic cable and power connection for the module.

The IO cabinet contains a power supply and an IO module. The IO module provides an interface between the fiber-optic receiver/transmitter board and the Cray Research 6-Mbyte/s channel.

At the FEI mainframe end (remote end of the FOL) of this link is an FEI cabinet. This cabinet consists of an FO cabinet positioned on top of an FEI cabinet. The FEI cabinet contains the modules necessary to communicate with the front-end computer system and a Cray Research IO module. The FO cabinet is identical to the FO cabinet at the local end of the link.

The electrical kit contains a Cray Research IO module, logic and power interconnections for the IO module, and logic and power interconnections for the signal connection to the FO module in the FO cabinet.



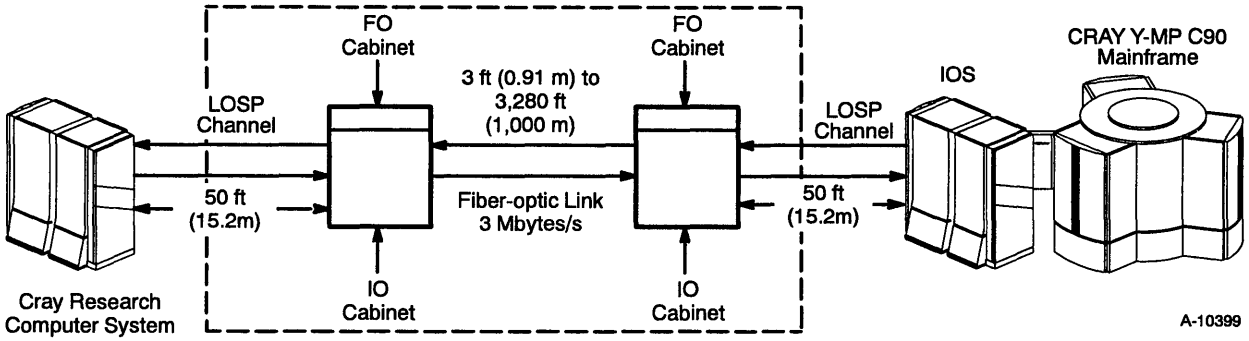
General FOL-3 Configuration

FOL-3 Specifications

The table to the right lists the required FOL-3 equipment. The number of kits the Cray Research customer needs to purchase varies for each Cray Research computer system depending on the site and system configuration.

The illustration below shows the general configuration of the FOL-3 when two Cray Research systems are configured together. The dotted line encircles the components that comprise the FOL-3.

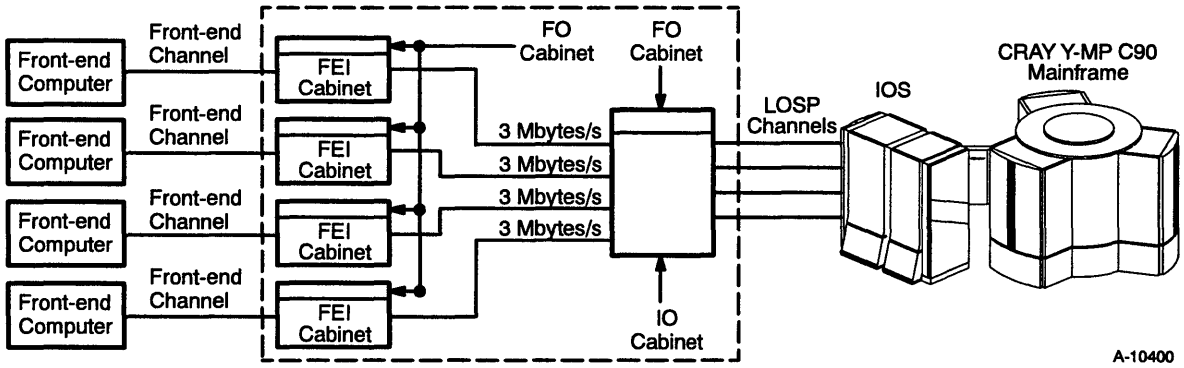
FOL-3 Equipment List	
Equipment	Quantity Needed
Electrical kit	One kit per FOL-3
FO cabinet	Two kits for initial installation; one kit thereafter
IO cabinet	One kit



FOL-3 Connection Between Two Cray Research Computers Systems

The illustration below shows the FOL-3 connected to a CRAY Y-MP C90 computer system and four front-end computers. The 6-Mbyte/s channel exiting the IOS connects to the IO interface cabinet. The fiber-optic cables exit the IO interface cabinet and are routed to the FEIs.

The FEIs are connected to the front-end computer by the front-end channel. The dotted line encircles the components that comprise the FOL-3.



FOL-3 Configured with Multiple Front-end Computer Systems

FOL-3 Specifications

The following additional fiber-optic cable configurations are possible for the FOL-3:

- 3-Mbyte/s, 4-km cable
- 6-Mbyte/s, 2-km cable

The equipment configurations for 2-km and 4-km cable lengths are identical to those for the FOL-3.

The customer is responsible for supplying and installing the fiber-optic cables. A variety of cable types and vendors exists. See your local Cray Research sales representative for cable specifications.

FOL-3 Advantages

The following items are advantages of using the FOL-3 as opposed to wire cabling:

- Decreased cost
- Increased security
- Increased cabling distances
- Decreased vulnerability to interference
- Ease of handling

FOL-3 Features

The table shown below describes FOL-3 features.

Physical Description

Floor space . . . . . 4.1 ft^2 (0.38 m^2)
Weight . . . . . 240 lbs (109 kg)
Height . . . . . 27 in. (69 cm)

FOL-3 Features

Feature	Description
Fiber-optic cable length	3 ft. (0.91 m) to 3,280 ft. (1,000 m)
Power Requirements	–5.2 V, –2.0 V, 100-W total power
Transfer Rate	3 Mbytes/s
Data Protection	Cyclic redundancy check (CRC) on link data, parity generation, and channel data check
Ground Isolation	Complete ground isolation between a Cray Research computer system and a front-end computer.

6 SOFTWARE OVERVIEW

CRAY Y-MP C90 computer systems come with a variety of software, including the Cray Research operating system UNICOS. The CF77 compiling system provides automatic vectorizing, as do the Cray Research Standard C and Pascal compilers. Extensive library routines, program- and file-management utilities, debugging aids, a powerful Cray Research assembly language, and extensive support for industry standards are included in the system software. A large number of third-party and public-domain application programs also run on Cray Research systems.

CRAY Y-MP C90 computer systems are supported by industry standard communications software such as the International Standards Organization Open Systems Interconnect (ISO/OSI) protocol and Transmission Control Protocol/Internet Protocol (TCP/IP). The CRAY Y-MP C90 systems are also supported by Cray Research proprietary station products for connecting to other vendors' systems and workstations.

UNICOS Operating System

CRAY Y-MP C90 computer systems come with the UNICOS operating system. The UNICOS operating system is derived from the UNIX Laboratories, Inc. UNIX System V operating system. It is also based in part on the Fourth Berkeley Software Distribution (BSD), under license from The Regents of the University of California.

The UNICOS operating system provides exceptional problem-solving ease; it provides powerful interactive and batch capabilities and multiple methods to accomplish a task. It efficiently manages high-speed data transfers between the CRAY Y-MP C90 system and peripheral equipment. The UNICOS operating system is written in C, a high-level language, and is available on all Cray Research systems.

The UNICOS operating system consists of a kernel plus a large set of utilities and library programs. The kernel is a simple 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 complex tasks.

The UNICOS operating system offers a large set of utility programs that allows the user to interact with the operating system. In addition, it provides a number of products specifically designed for Cray Research computer systems. The UNICOS operating system supports the following compilers: Fortran, Pascal, C, Lisp, and Ada.

The UNICOS operating system and UNIX are essentially the same in philosophy, structure, and function. However, Cray Research has enhanced UNIX to create the UNICOS operating system to use the power of the Cray Research computer system more efficiently. Enhancements include I/O capabilities to take advantage of supercomputer performance, added multiprocessor and multitasking support, additional networking software, accounting features, and others. The UNICOS operating system is designed for both interactive and batch environments. It supports the Network Queuing System (NQS) for batch processing.

Multiprocessing

Multiprocessing divides an application program into independent tasks called partitions and runs them in parallel. Compared to serially executed programs, multiprocessing can substantially improve throughput. Three multiprocessing features have evolved, and they can all work together in a single program, but not in the same program unit. The following subsections describe the three multiprocessing features.

Macrotasking Feature

Macrotasking is the first phase in the evolution of Cray Research parallel processing software. It requires extensive data scoping and insertion of Cray Research-specific library calls that allow parallel execution of code at the subroutine level on multiple processors. Macrotasking is best suited to programs with large, long-running tasks. The user interface to the system's macrotasking capability is a set of Fortran-callable subroutines that explicitly define and synchronize tasks at the subroutine level. These subroutines are compatible with similar subroutines available on other Cray Research products.

Microtasking Feature

Microtasking is the second phase in the evolution of Cray Research parallel processing software. Microtasking expands on the strengths of macrotasking but requires less data scoping and uses compiler directories rather than Cray Research-specific library calls.

Microtasking is a multiprocessing technique that allows parallel execution of very small segments of code on multiple processors. An example of this is individual iterations of DO loops. With microtasking, the programmer can revise the code or issue 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. Additionally, in a batch environment where processors may become available for short periods, the microtasked job can dynamically adjust to the number of available processors.

Autotasking Feature

The Autotasking feature (Autotasking) of the CF77 Fortran compiling system is the third phase in the evolution of Cray Research parallel processing software. Autotasking is based on the microtasking design and shares several advantages with microtasking: very low overhead synchronization cost, excellent dynamic performance independent of the number of central processing units (CPUs) available, both large and small granularity parallelism, and so on.

Autotasking has two fundamental improvements over microtasking. First, it is automatic multiprocessing. Autotasking allows user programs to be automatically partitioned over multiple CPUs (without user intervention). Second, Autotasking can exploit parallelism at the DO-loop level without extending to subroutine boundaries.

CF77 Compiling System

CRAY Y-MP C90 computer systems use the Cray Research CF77 compiling system. This compiling system is fully compliant with the ANSI 78 (Fortran 77) standards and offers a high degree of automatic scalar and vector optimization. The CF77 compiling system permits maximum portability of programs between different Cray Research systems and accepts many nonstandard programming constructs written for other vendors' compilers. Vectorized object code is produced from standard Fortran code; users can program in standard syntax to access the full power of the mainframe architecture.

The CF77 compiling system consists of the following software: the Autotasking Fortran Dependence analyzer, the Fortran translator, and the Cray Research Fortran 77 compiler (CFT77). This system is a multipass, optimizing, transportable compiling system that processes existing

standard Fortran programs. It uses two basic techniques to improve the execution time of a Fortran program: vectorization and scalar optimization.

The compiling system automatically generates code that uses the vector registers and functional units of the mainframe. The programmer does not need to know the details of vectorization because the compiling system automatically vectorizes Fortran programs. When the compiling system 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.

The CF77 compiling system is portable on several levels. Because it is in compliance with the ANSI 78 standard, programs written for other computer systems have maximal portability to a CRAY Y-MP C90 system with minimal effort. Also, the compiling system is designed to run on all Cray Research systems enabling a Fortran program that compiles and runs on one Cray Research system to run on all Cray Research systems. In general, programs that compile and execute correctly with the CFT compiler also compile and execute correctly with the CF77 compiling system.

C Compiler

The C language is a high-level system programming language. Most of the UNICOS kernel code and utilities are written in C because C is a structured and highly efficient language. Many programming applications are also written in C. The C language 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 Cray Research Standard C compiler performs scalar optimization and vectorizes code automatically.

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

The C preprocessor (cpp) is included as a part of the Cray Research Standard C compiler. The cpp enables macro substitution, conditional compilation, and the inclusion of named files in the compilation process.

Cray Research Standard C is portable on several levels. Because Cray Research Standard C is in compliance with the 1989 ANSI standard, programs written for other computer systems have maximal portability to a CRAY Y-MP C90 system with minimal effort.

Pascal

Pascal is a high-level, general-purpose programming language used as the implementation language for the CF77 compiling system and other Cray Research products. Cray Research 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 execute on Cray Research computer systems. Using Pascal, a programmer can implement algorithms and data structures in a high-level, machine-independent manner without sacrificing efficiency.

The Cray Research Pascal compiler takes advantage of the mainframe 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 Assembly Language (CAL) enables a user to closely tailor a program to the architecture of the mainframe. Through CAL, a programmer may symbolically express all hardware functions of the mainframe. CAL allows the production of highly efficient machine language programs. The user may designate program and data information to enable complete control of the mainframe CPUs. This facilitates full use of various features, such as the shared text feature, whereby a single set of instructions can service many users simultaneously.

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.

Cray Ada Environment

The Cray Ada Environment includes a Cray Ada compiler and a set of related tools that link, debug, and maintain Ada application programs. The Cray Ada Environment also supports an implementation of Ada program libraries, providing for flexible, project-oriented software development. The Cray Ada Environment is validated under the current Ada Compiler Validation Capability (ACVC) test suite.

Cray Allegro CL

Cray Allegro CL is a complete implementation of Common Lisp. The Cray Allegro CL system consists of an interpreter, an optimizing compiler, and a set of functions. There are a number of extensions to the specification in Cray Allegro CL. Included among the extensions are the top level, the debugger, a foreign function interface, and Flavors, an object-oriented programming system. Cray Allegro CL was designed to be compact, fast, and robust with respect to detecting user errors. The implementation itself is written mostly in Common Lisp, with some portions written in the C language.

Subroutine Libraries

Cray Research software includes subroutines that are callable from the CF77 compiling system, C, Pascal, and CAL. The subroutines are divided into libraries, generally on a functional basis. Libraries containing various utilities, high performance I/O subroutines, and numerous math and scientific routines are available, as are special-purpose libraries.

Utilities

A broad variety of software tools assists both interactive and batch users in the efficient use of a CRAY Y-MP C90 computer system.

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

The Cray Symbolic Debugger (CDBX) allows users to interactively detect program errors by examining both running programs and program memory dumps. Other debugging tools are available for dump analysis and interpretation.

A variety of performance aids assists in analyzing program performance and optimizing programs with minimal effort. These aids include both static and dynamic analyzers, as well as profilers for CPU and I/O usage. Many provide graphic user interfaces using the X Window System.

The UNICOS Source Manager utility tracks modifications to files. This system is useful when programs and documentation undergo frequent changes because of development, maintenance, or enhancement. Line- and screen-oriented text editors, such as vi and Emac, offer versatility for users who wish to create and maintain text files. Other system utilities provide for proper management of the system resources.

Communications Software

A CRAY Y-MP C90 computer system fits into environments consisting of single or multiple Cray Research systems, other vendors' mainframes, minicomputers, workstations, and devices capable of high-speed data transfer. Cray Research provides easy user access to Cray Research system capabilities, the ability to distribute applications between Cray Research computer systems and other vendor systems, and effective integration into existing customer networks.

Communications and connectivity are supported by the Transmission Control Protocol/Internet Protocol (TCP/IP) suite, the International Standards Organization Open Systems Interconnect (ISO/OSI) protocol, and the UNICOS station call processor (USCP) protocol.

The TCP/IP product allows the CRAY Y-MP C90 computer system to function as a peer in TCP/IP-supported, open networking environments. TCP/IP is a set of computer networking protocols that enables two or more hosts to communicate. Further, it is a set of procedures that allows communication among 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 only under the UNICOS operating system.

The ISO/OSI protocol logically connects Cray Research systems to other systems running ISO/OSI protocols. The UNICOS operating system supports the File Transfers, Access, and Management (FTAM) and Virtual Terminal (VT) applications of OSI. FTAM provides an interactive file transfer service for unstructured text files, unstructured binary files, and file directory files. VT allows users to connect to a remote system from a Cray Research system and to use the resources of the remote system.

USCP provides, by way of station software products, support for communicating with various vendor systems through a vendor's proprietary networking capability, such as IBM's SNA or DEC's DECnet.

Cray Research station software products provide system access to proprietary protocol implementations through network gateways. Cray Research 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 Research systems.

The following stations are available:

- Apollo station - provides the software connection between the Cray Research mainframe and the Domain workstation.
- CYBER station - joins the Cray Research system to the Control Data Corporation CYBER 180 series, 70/170, or 700/800 systems to form a powerful computing combination.
- VAX or VMS station - controls the hardware and software link between a DEC VAX computer system and a Cray Research computer system.
- MVS station - provides the software connection between an IBM System/370, Extended Architecture, or compatible computer system and a Cray Research computer system.
- 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 Research computer system.
- UNIX station - provides Cray Research operating system access to installations whose front ends run UNIX.
- SUPERLINK/MVS product - provides data access, application-to-application communication, and job processing between the UNICOS operating system and MVS systems.
- CLS-UX product - provides Cray Research operating system access to UNIX users through a VAX or VMS system.

Applications

Cray Research supports application software vendors in converting and optimizing software for CRAY Y-MP C90 computer systems. Many of the most widely used application programs are currently available and supported to run in the Cray Research UNICOS environment. 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.

Cray Research has also developed the UniChem and MPGS applications. UniChem is an integrated software environment for chemists. MPGS is an interactive postprocessing visualization tool.

The availability of applications for Cray Research systems is driven largely by customer requirements that are communicated to the software vendors. Cray Research supports the on-going process of converting and maintaining application software.

Software Publications

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

UNICOS Operating System

Publication Number	Title
SG-2005	I/O Subsystem (IOS) Operator's Guide for UNICOS
SG-2017	UNICOS Source Code Control System (SCCS) User's Guide
SG-2050	UNICOS Text Editors Primer
SG-2052	UNICOS Overview for Users
SG-2112	UNICOS Installation Guide
SG-2113	UNICOS System Administration for Source Releases
SR-2011	UNICOS User Commands Reference Manual
SR-2012	Volume 4: UNICOS System Calls Reference Manual
SR-2014	UNICOS File Formats and Special Files Reference Manual
SR-2022	UNICOS Administrator Commands Reference Manual

Fortran

Publication Number	Title
SR-3071	CF77 Compiling System, Volume 1: Fortran Reference Manual

C

Publication Number	Title
SR-2074	Cray Standard C Programmer's Reference Manual

Pascal

Publication Number	Title
SR-0060	Pascal Reference Manual

Libraries

Publication Number	Title
SR-2057	Volume 5: UNICOS Network Library Reference Manual
SR-2079	Volume 1: UNICOS Fortran Library Reference Manual
SR-2080	Volume 2: UNICOS Standard C Library Reference Manual
SR-2081	Volume 3: UNICOS Math and Scientific Library Reference Manual

Utilities

Publication Number	Title
SD-2107	I/O Subsystem Model E (IOS-E) Guide
SG-2051	UNICOS Tape Subsystem User's Guide
SG-2094	UNICOS CDBX Debugger User's Guide
SG-3074	CF77 Compiling System, Volume 4: Parallel Processing Guide
SG-3078	OWS-E Operator Workstation Operator's Guide
SG-3079	OWS-E Operator Workstation Administrator's Guide
SR-0010	Software Tools Reference Manual
SR-0066	Segment Loader (SEGLDR) and **ld** Reference Manual
SR-2091	UNICOS CDBX Symbolic Debugger Reference Manual
SR-3077	OWS-E Operator Workstation Reference Manual

Communications Software

Publication Number	Title
SA-0250	Apollo DOMAIN Station Reference Manual
SC-0270	CDC NOS/VE Link Software Command Reference Manual
SG-2009	TCP/IP and OSI Network User's Guide
SI-0038	IBM MVS Station Reference Manual
SI-0160	IBM VM Station Command and Reference for COS
SI-0191	SUPERLINK Guides

Publication Number	Title
SR-0034	CDC NOS/BE Station Reference Manual
SR-0035	CDC NOS Station Reference Manual
SU-0107	UNIX Station User's Guide
SU-3121	CLS-UX Station User Guide
SV-0020	DEC VAX/VMS Station Reference Manual

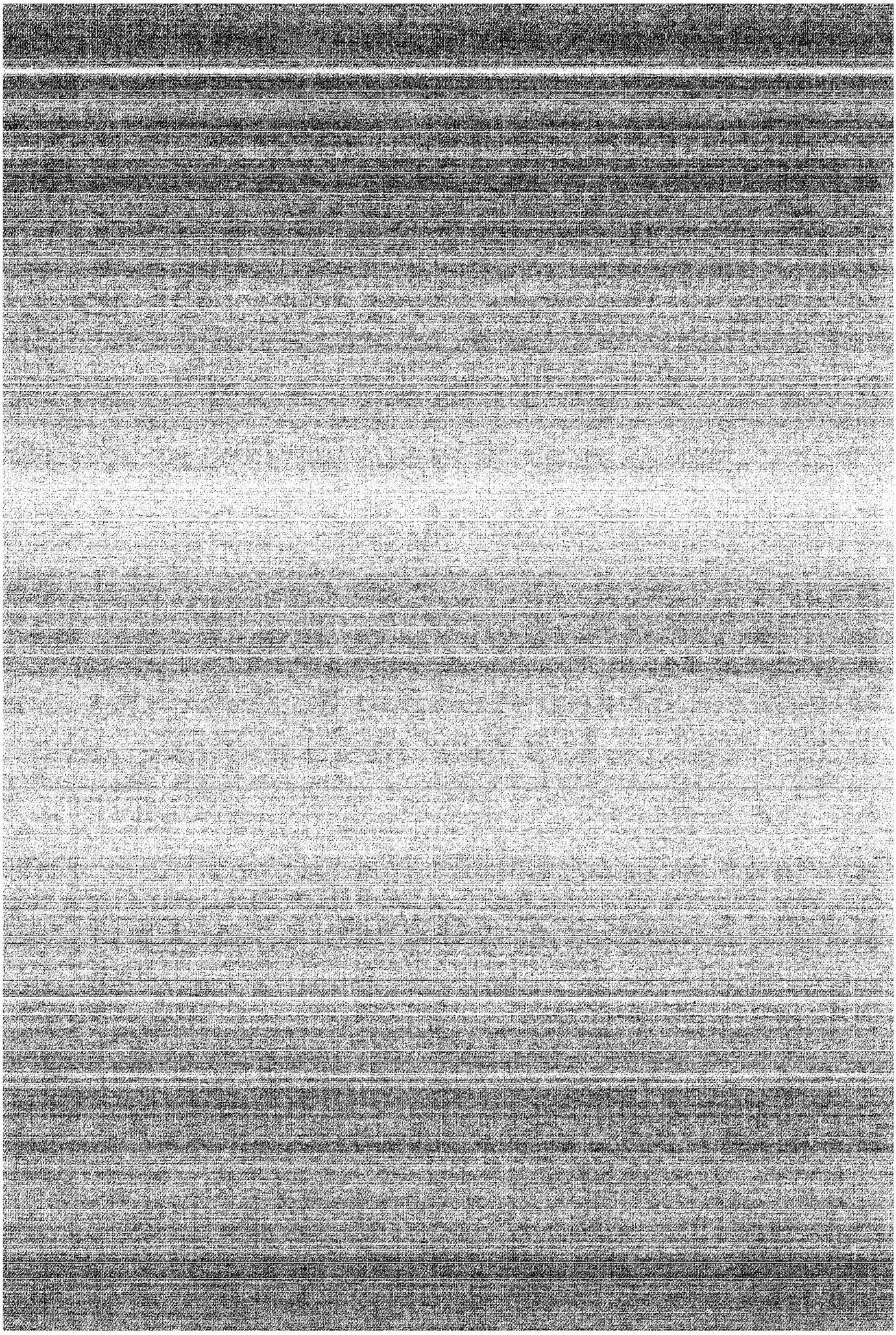
Applications

Publication Number	Title
MCDR-1000M	Directory of Applications Software for Cray Research Supercomputers for 1992

Software Training

Cray Research offers complete training on the software available for CRAY Y-MP C90 computer systems. Extensive user-support analyst and system analyst training is available at Cray Research's training facility. End-user and operator training are available at customer sites after installation of a Cray Research computer system. More information regarding courses and schedules can be obtained through your local or regional Cray Research sales office.

GLOSSARY



GLOSSARY

A

Application Software designed to perform a particular job or set of related jobs.

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

Autotasking The process of automatically dividing up a program into individual tasks and organizing them to make the most efficient use of the Cray Research, Inc. computer hardware; a trademark of Cray Research, Inc.

Auxiliary input/output processor (EIOP) A quarter board in the IOS-E that controls the transfer of data between the channel adapters (CAs) and the IOS-E buffer board.

B

Bank The smallest addressable division of central memory.

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

BDM Bidirectional memory mode (bit). The modes field in the exchange package contains the BDM mode bit. When the BDM mode bit is set, block read and write operations can operate concurrently.

BiCMOS Bipolar complementary metal oxide semiconductor.

BPI Breakpoint interrupt (flag). The breakpoint interrupt flag sets if the interrupt-on-breakpoint (IBP) interrupt mode bit is set and enabled and a write reference is made to an address within the breakpoint range.

Buffer board A module in the IOS-E that temporarily stores system data from the high-speed (HISP) channels or the channel adapters (CAs).

C

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 pseudoinstructions.

CCA-1 A channel adapter that connects the IOS-E to a 6-Mbyte/s channel pair.

Central memory Memory residing in the mainframe.

Central processing unit (CPU) A module used in the mainframe that controls the flow of system data, performs mathematical and logical functions on system data, and executes program instructions.

Chaining The process of sequencing logical operations so the results of one operation may be used by another operation without needing a memory reference in between.

Channel adapter (CA) A component in the IOS-E that transfers control and data between the buffer board and the peripherals.

Check bits Bits used to determine if one or more bits in a word have an incorrect value.

CLN Cluster number (register). The CLN register in the exchange package determines which set of the 17_{10} available clusters of SB, ST, and SM registers the CPU can access.

Clusters A set of shared registers accessible by all CPUs. There are 17_{10} valid clusters of shared registers in a CPU.

Cluster interface (CIN) A quarter board in the IOS-E that transfers control and data between the workstation interface (WIN) and the input/output processor multiplexer (IOP MUX) and auxiliary input/output processors (EIOPs).

Compiler A software program used to convert high-level programming language into binary machine code.

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

CRI Cray Research, Inc.

C90 Mode The modes field in the exchange package contains the C90 mode bit. When the C90 mode bit is set, the mainframe operates in C90 mode. The V registers are 64 bits x 128 elements, the VL register is 8 bits wide, and the P register is 32 bits wide, which enables a program range of 1 Gword. The entire CRAY Y-MP C90 instruction set is also available.

D

DCA-1 A channel adapter that connects the IOS-E to DS-40, DS-41, and DS-49 disk storage units.

DCA-2 A channel adapter that connects the IOS-E to DD-60 and DD-61 disk drives. The DCA-2 disk channel adapter in the IOP manages control signals and protocol for the individual disk drives in a DE-60.

DBA Data base address (register). The DBA register, part of the exchange package, holds the base address of the user's data range.

DCC-2 The Cray Research DCC-2 houses the DC-40, which is separate from the DD-40 disk drives.

DCU Disk controller unit. An interface between the disk storage units (DSUs) and the auxiliary I/O processor (EIOP).

DC-40 The Cray Research DC-40 disk controller provides interface logic to adapt DCA-1 signals and protocol for individual DS-40s, to handle routing among the drives, and to buffer data from the four spindles in a full-track buffer.

DC-41 The Cray Research DC-41 disk controller provides interface logic to adapt DCA-1 signals and protocol for individual disk drive units, to handle routing among the drives, and to buffer data from the four spindles in a full-track buffer.

DD-40 The Cray Research DD-40 disk drive.

DD-41 The Cray Research DD-41 disk drive.

DD-49 The Cray Research DD-49 disk drive.

DD-60 The Cray Research DD-60 high-performance disk drive.

DD-61 The Cray Research DD-61 high-performance disk drive.

DE-60 One Cray Research DE-60 disk enclosure cabinet contains a maximum of nine DD-60 disk drives. Eight of the disk drives store system data, and the ninth disk drive is a spare. The DCA-2 disk channel adapter in the IOP manages control signals and protocol for the individual disk drives in a DE-60.

Deadlock A condition 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.

D (continued)

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

Dielectric coolant A fluid that travels through the module cold plates, removes heat from the modules, and transfers the heat to refrigerant in the heat exchange subassembly.

DL Deadlock interrupt (flag). The deadlock interrupt flag sets if the IDL interrupt mode bit is set, the program is not in monitor mode, and a deadlock condition occurs because all CPUs in a cluster are holding issue on a test and set instruction.

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

DRAM Dynamic random access memory. A memory device that must be refreshed periodically in order to store data.

DSU Disk storage units. A computer disk drive.

DS-40 The DS-40 disk subsystem consists of the DD-40 disk drive, the DC-40 disk control unit (DCU), and the disk controller cabinet (DCC-2).

DS-41 The DS-41 disk subsystem consists of the the DC-41 disk controller and the DD-41 disk drive.

E

EIM Enable interrupt modes (flag). The interrupt modes field in the exchange package contains the EIM flag. An exchange to monitor mode or non-monitor mode sets the EIM flag.

EIOP Auxiliary I/O processor. The EIOP controls the channel adapters that connect the IOS-E to peripheral devices such as disk drives, tape drives, and communications channels.

EMI Electromagnetic interference. EMI is radiated energy that interferes with and distorts digital signals.

ESL Enable second vector logical mode (bit). The modes field in the exchange package contains the ESL mode bit. When the ESL mode bit is set, the second vector logical functional unit is enabled, and if it is not busy, it has first priority to execute instructions 140*ijk* through 145*ijk*.

Ethernet A particular type of network hardware that forms a physical link between computers; a trademark of Xerox Corporation.

E (continued)

Exchange mechanism The technique used in the CRAY Y-MP C90 computer system for switching instruction execution from program to program. Refer to exchange package.

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.

EXX Error exit interrupt (flag). The EXX interrupt flag sets if the enable-interrupt-on-error-exit (FEX) interrupt mode bit is set and enabled and an error exit instruction (000000) issues. Issuing an error exit instruction always causes an exchange, regardless of the state of FEX.

F

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 Front-end interface. An interface that connects the CRAY Y-MP C90 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.

Fetch sequence A fetch sequence transfers a block of instructions from memory to an instruction buffer.

FEX Enable-interrupt-on error exit Mode (bit). The interrupt modes field in the exchange package contains the FEX bit. When the FEX bit is set, it enables an interrupt when an error exit occurs.

Floating-point operation A mathematical or logical operation on two or more real numbers.

Fluorinert Liquid The dielectric coolant circulated through the module cold plates; a trademark of 3M.

FNX Interrupt-on-normal exit mode (bit). The interrupt modes field in the exchange package contains the FNX bit. When the FNX bit is set, it enables the NEX interrupt flag to set if a normal exit instruction (004000) issues. Issuing a normal exit instruction always causes an exchange, regardless of the state of FNX. This mode is not affected by the EIM flag.

F (continued)

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

FPE Floating-point error (flag). The interrupt flags field 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 Interrupt-on-floating-point error (IFP) flag is set.

FPS Floating-point error status (bit). The status field in the exchange package contains the FPS status bit. The floating-point status bit sets if a floating-point error occurred during the execution interval.

Functional unit Circuitry designed to perform a particular mathematical or logical operation.

G

Gate array An array of circuits contained in a single integrated circuit package; these circuits may be customized in their operation as a group.

Gather/scatter An operation that places data at various intervals in the available memory storage and then gathers the data back into its original organization.

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.

HCA-3 A channel adapter that connect the IOS-E to a HIPPI input channel.

HCA-4 A channel adapter that connects the IOS-E to a HIPPI output channel.

HDA Head disk assembly. A sealed assembly that contains the magnetic storage media (disk drive), read and write heads, and servo mechanism.

HEU Heat exchange unit. Part of the cooling equipment for the CRAY Y-MP C90 mainframe. The HEU uses a refrigerant to cool the dielectric coolant that circulates through the mainframe.

High Performance Parallel Interface (HIPPI) A type of interface used to transfer control and data between Cray Research, Inc. channel adapters and peripherals.

H (continued)

High-speed (HISP) channel A channel that transfers system data between the IOS-E and the mainframe or between the IOS-E and the SSD-E.

High-speed control multiplexer (HCM) A quarter board in the IOS-E that controls the transfer of high-speed (HISP) channel information between the IOS-E and the SSD-E or mainframe.

HYPERchannel A trademark and product of Network Systems Corporation (NSC) that interfaces a LOSP channel to other brands of computers.

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.

IBP Interrupt-on-breakpoint mode (bit). The interrupt modes field in the exchange package contains the IBP bit. When the IBP bit is set, it enables an interrupt if a breakpoint occurs.

ICM Interrupt-on-correctable memory error mode (bit). The interrupt modes field in the exchange package contains the ICM bit. When the ICM bit is set, it enables interrupts on correctable memory data errors while data is being read from memory.

ICP Interprocessor Interrupt (flag). The interprocessor interrupt flag sets if the IIP interrupt mode bit is set and enabled and another CPU requests an interrupt of this CPU by issuing instruction 0014*j*1.

IDL Interrupt-on-deadlock mode (bit). The interrupt modes field in the exchange package contains the IDL bit. When the IDL bit is set, it enables an interrupt if a deadlock occurs while the program is not in monitor mode. IDL has no effect in monitor mode.

IFP Interrupt-on-floating-point error mode (bit). The interrupt modes field in the exchange package contains the IFP bit. When the IFP bit is set, it enables interrupts on floating-point errors.

IIO Interrupt-on-I/O mode (bit). The interrupt modes field in the exchange package contains the IIO bit. When the IIO bit is set, it enables an interrupt if SIE is set and this CPU is the lowest-numbered CPU with IIO=1 and EIM=1.

I (continued)

IIP Interrupt-on-interprocessor interrupt mode (bit). The interrupt modes field in the exchange package contains the IIP bit. When the IIP bit is set, it enables an interprocessor interrupt if requested by another CPU.

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.

IMC Interrupt-on-request from MCU mode (bit). The interrupt modes field in the exchange package contains the IMC bit. When the IMC bit is set, it enables interrupts from the MCU.

IMI Interrupt-on-monitor mode instruction mode (bit). The interrupt modes field in the exchange package contains the IMI bit. When the IMI bit is set, it enables an interrupt if a monitor mode instruction (001*ijk*; $j \neq 0$) issues while the program is not in monitor mode. IMI has no effect in monitor mode.

Input/output cluster (IOC) A component of the IOS-E that contains one cluster interface (CIN), one input/output processor multiplexer (IOP MUX), four auxiliary input/output processors (EIOPs), two high-speed control multiplexers (HCMs), one buffer board, sixteen channel adapters (CAs), and one programmable interrupt (PINT).

Input/output processor multiplexer (IOP MUX) A quarter board in the IOS-E that transfers information between the high-speed control multiplexers (HCMs), the cluster interface (CIN), and the auxiliary input/output processors (EIOPs).

Input/output subsystem model E (IOS-E) A component of a CRAY Y-MP C90 computer system that transfers system data between the peripherals, SSD solid-state storage device model E (SSD-E), and the mainframe.

Intelligent peripheral interface - 2 (IPI-2) An interface used to transfer control and system data between Cray Research, Inc. channel adapters and peripherals.

Interrupt modes field The interrupt modes field in the exchange package contains user-selectable bits that dictate the execution of the program.

Instruction buffer A set of registers in a CRAY Y-MP C90 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.

I (continued)

Instruction set A set of instructions that a particular computer can perform.

I/O buffer A buffer used to provide temporary storage for data transferred between the mainframe and peripheral devices.

IOC Refer to input/output cluster.

IOI I/O interrupt (flag). The I/O interrupt flag sets if the SIE bit is set and this CPU is the lowest-numbered CPU with IIO interrupt mode set and enabled when a LOSP or VHISP channel completes a transfer.

IOP I/O processor. An IOP is a fast, multipurpose computer capable of transferring data at extremely high rates. The IOS-E contains multiple IOPs.

IOR Operand range error mode (bit). The interrupt modes field in the exchange package contains the IOR bit. When the IOR bit is set, it enables interrupts on operand address range errors.

IPC Interrupt-on-request from programmable clock mode (bit). The interrupt modes field in the exchange package contains the IPC bit. When the IPC bit is set, it enables an interrupt on a request from the programmable clock.

IPI-2 An interface used to transfer control and system data between Cray Research, Inc. channel adapters and peripherals.

IPR Interrupt-on-program range error mode (bit). The interrupt modes field in the exchange package contains the IPR bit. When the IPR bit is set, it enables the PRE interrupt flag to set if a program range error occurs.

IRP Interrupt-on-register parity error mode (bit). The interrupt modes field in the exchange package contains the IRP bit. When the IRP bit is set, it enables an interrupt if a register parity error is detected while data is being read from a register.

IRT Interrupt-on-request from real-time clock mode (bit). The interrupt modes field in the exchange package contains the IRT bit. When the IRT bit is set, it enables an interrupt on a request from the real-time clock.

Issue sequence The issue sequence selects the instruction indicated by the program address (P) register, decodes it, determines whether the required registers or functional units are available, and if so, enables the CPU to execute the instruction.

I (continued)

IUM Interrupt-on-uncorrectable memory error mode (bit). The interrupt modes field in the exchange package contains the IUM bit. When the IUM bit is set, it enables interrupts on uncorrectable memory data errors.

L

Library A set of commonly used software routines that are available to programmers and to programs being compiled.

LLRC Length/longitudinal redundancy check. An error control system based on the arrangement of data in blocks according to some preset rule, the correctness of each character within the block being determined on the basis of the specific rule or set.

Low-speed (LOSP) channel Low-speed channel. The LOSP channel has a transfer rate of 6 or 20 Mbytes/s and enables an I/O cluster to communicate with a mainframe, front-end interface, or SSD-E.

M

Mainframe A component of a CRAY Y-MP C90 computer system that contains central memory and central processing units (CPUs).

Maintenance channel A channel that connects the MWS-E to the SSD-E.

Maintenance control unit interface (MCUI) A quarter board in the mainframe that receives an interrupt signal from the programmable interrupt (PINT) and interrupts the specified CPU in the mainframe.

Maintenance workstation model E (MWS-E) A component of a CRAY Y-MP C90 computer system that provides an intelligent and dedicated platform for performing offline and online tests, monitoring environmental conditions, and recording hardware errors.

MCU Maintenance control unit. The maintenance control unit for the CRAY Y-MP C90 in the MWS-E.

MCU Maintenance control unit interrupt (flag). The MCU interrupt flag sets if the IMC interrupt mode bit is set and enabled and the MCU interrupt signal becomes active on I/O channel 40.

MGS Motor-generator set. An MGS converts primary power from commercial power mains to the voltage and frequency used by the mainframe power supplies.

M (continued)

MEC Memory error correctable interrupt (flag). The memory error (correctable) flag sets if the ICM interrupt mode bit is set and a correctable memory error occurs while data is being read from memory.

MEU Memory error uncorrectable (flag). The memory error (uncorrectable) flag sets if IUM interrupt mode is set and enabled and an uncorrectable memory error occurs while data is being read from memory.

MFC Mainframe chassis.

MII Monitor instruction interrupt (flag). The monitor instruction interrupt flag sets if the IMI interrupt mode bit is set and a monitor mode instruction (001*ijk*; $j \neq 0$) issues while the program is not in monitor mode.

MM Monitor mode (bit). The modes field in the exchange package contains the MM bit. When the MM mode bit is set, it inhibits all interrupts except memory errors, normal exit, and error exit. The program can execute those instructions that are privileged to monitor mode.

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

Multitasking The capability to run two or more parts, or tasks, of a single program in parallel on different CPUs within a mainframe.

Multiprocessing Several computer processes or jobs being computed at the same time.

Multiprogramming The process of writing software to utilize the capabilities of a computer to process multiple jobs simultaneously.

N

NEX Normal exit interrupt (flag). The interrupt flags field in the exchange package contains the NEX flag. The normal exit flag sets if the interrupt-on-normal-exit (FNX) interrupt mode bit is set and enabled and a normal exit instruction (00400) issues. Issuing a normal exit instruction always causes an exchange, regardless of the state of FNX.

O

ORE Operand range error (flag). The interrupt flags field 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 bit is set.

Operating system The major controlling software running in a computer that controls its overall operation.

Operator workstation model E (OWS-E) A component of a CRAY Y-MP C90 computer system that Cray Research, Inc. analysts and customers use to monitor the computer system.

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.

Pascal A high-level programming language.

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

Parity Equivalence in the check bit of transmitted and received data.

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 P register is 24 bits wide in Y-MP mode and 32 bits wide in C90 mode.

PCI Programmable clock interrupt (flag). The programmable clock interrupt flag sets if the IPC interrupt mode bit is set and enabled and the counter in the programmable clock equals 0.

Pipelining An operation or instruction that begins before a previous operation or instruction finishes. Pipelining is accomplished using fully segmented hardware.

PN Processor number. The PN field in an exchange package indicates which CPU executed the exchange sequence.

Port A hardware or software access path to memory.

P (continued)

PRE Program range error (flag). The interrupt flags field 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.

Programmable interrupt (PINT) A quarter board in the IOS-E that enables any input/output cluster (IOC) in the IOS-E to interrupt any CPU in the mainframe.

Protocol Software that defines the precise way in which data is transferred from one place to another.

PS Program status (bit). The interrupt modes field in the exchange package contains the PS bit. The PS bit is set by the operating system to denote whether a CPU concurrently processing a program with another CPU is the master or slave in a multitasking situation.

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.

RAM Random access memory. A memory device that retains the stored data as long as power is applied. When power is removed from the device, the stored information is lost.

RCU Refrigeration and condensing unit. The RCUs dissipate the heat transferred from the heat exchanger units (HEUs).

Reciprocal approximation The mathematical process of approximating the value of a real number when divided into one (1/n).

Refrigerant A fluid that removes heat from dielectric coolant and transfers the heat to water or air.

Register A hardware storage location for one word, byte, or element of data.

R (continued)

RISC Reduced instruction set computer.

RPE Register parity error (flag). When a word is written into a register, a set of parity bits is generated and stored with the data bits. This set of parity bits is compared to another set that is generated when the word is read out of the register. An error is indicated when the two sets do not match. Parity errors set the register parity error (RPE) flag in the exchange package.

RT Load real-time clock (instruction). The RT instruction loads the real-time clock register with the contents of a scalar (S) register.

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

RTI Real-time interrupt (flag). The real-time interrupt flag sets if the IRT interrupt mode bit is set and enabled and a real-time interrupt request is received.

S

Scalar A single numerical value that represents a single aspect of a physical quantity.

Section A major addressable division of central memory that may be further divided into subsections and banks.

Semaphore A 1-bit value stored in a register and used by programs to communicate the occurrence of an event.

Shared registers Registers that are available for more than one CPU to write to and read from.

Single-byte correction/ double-byte detection (SBCDBD) A method of detecting whether one or more 4-bit bytes in a word has an incorrect value. If only one byte has an incorrect value, that byte can be changed back to the correct value.

Single-error correction/double-error detection (SECDED) A method of detecting whether one or more bits in a word has an incorrect value. If only one bit has an incorrect value, that bit can be changed back to the correct value.

Small computer system interface (SCSI) devices Components of the MWS-E and OWS-E that store and retrieve data used in the workstations.

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

S (continued)

SB Shared address (register). The SB register is a shared register used for transferring address information from one CPU to another.

Segmentation An operation that is divided into a discrete number of sequential steps, or segments. Fully segmented hardware is designed to perform one segment of an operation during a single clock period (CP).

SEI Selected for external interrupts (flag). The interrupt modes field in the exchange package contains the SEI flag. When the SEI flag is set, this CPU is preferred for I/O interrupts.

Shared registers Registers that are available for more than one CPU to write to and read from.

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

Source code A software program written in a high-level programming language.

Spindle A component of a DD-40 or DD-41 disk drive.

SPARC Scaleable Processor ARChitecture. A trademark of SPARC International.

SSD solid-state storage device model E (SSD-E) A component in a CRAY Y-MP C90 computer system that provides secondary data storage for the IOS-E and the mainframe.

ST Shared T (register). The ST register is a shared register used for transferring data from one CPU to another.

Status field The exchange package contains a status field that is used to determine the operating modes of a CPU.

Subsection A major addressable division of memory that may be further divided into banks.

System Maintenance and Remote Testing Environment (SMARTE) A Cray Research trademark; an online program that performs hardware verification, error detection, error isolation, and automated degradation of faulty hardware components.

T

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

TCA-1 Tape subsystem channel adapter.

U

UNICOS An operating system for Cray Research computer systems based primarily on the UNIX System Laboratories, Inc. UNIX System V and partially on the Fourth Berkeley Software Distribution. UNICOS is essentially the same in philosophy, structure, and function as UNIX, but has been enhanced to exploit the power of Cray Research computer systems.

V

Vector A single numerical value that contains information on more than one aspect of a physical quantity.

Very high-speed (VHISP) channel A channel that transfers system data between the mainframe and the SSD-E.

V register Vector register. Each V register contains 64 bits x 64 elements in Y-MP mode and 64 bits x 128 elements in C90 mode.

VL Vector length (register). The program-selectable VL register controls the effective length of a vector register for any operation. The VL register is 7 bits wide in Y-MP mode and 8 bits wide in C90 mode.

VM Vector mask (register). The VM field allows for the logical selection of particular elements of a vector.

VNU Vector not used status (bit). The state of the VNU bit in the exchange package status field indicates whether vector instructions (077xxx or 140xxx through 177xxx) were issued during the execution interval.

W

Warning and control system (WACS) A system that monitors the refrigeration and power distribution systems to ensure that the computer system is operating within recommended temperature and voltage limits.

Word A set amount of data that contains 64 bits of system data and 8 check bits.

Workstation interface (WIN) A quarter board in the IOS-E that transfers control and data between the MWS-E or OWS-E and the cluster interface (CIN).

WS Waiting for semaphore status (bit). The interrupt modes field in the exchange package contains the WS status bit. The waiting on semaphore bit sets if a test and set instruction (0034*jk*) is holding issue.

X

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

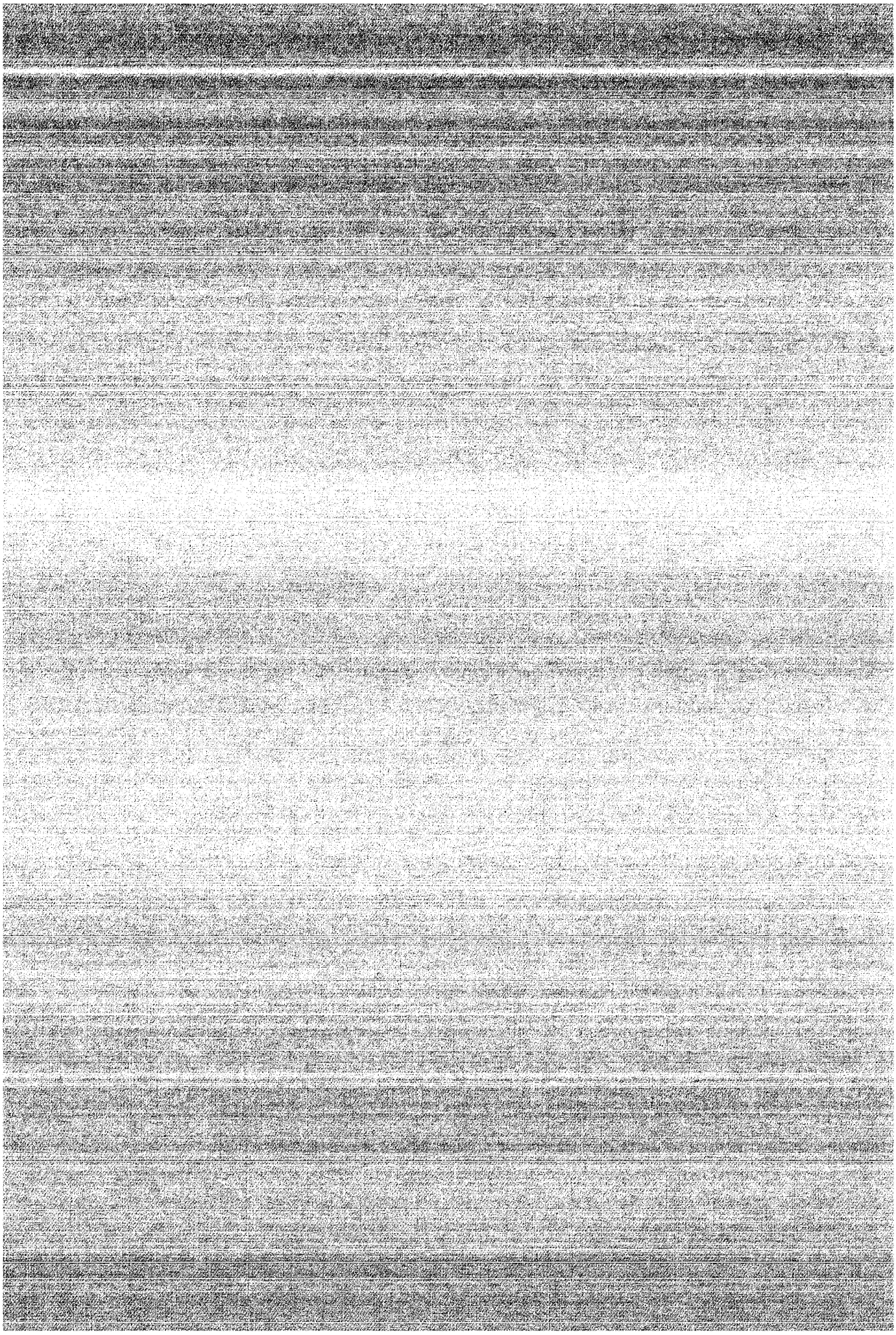
Y

Y-MP mode Y-MP mode is selected by setting the C90 mode bit in the exchange package to 0. When the mainframe is operating in Y-MP mode, the V registers are 64 bits x 64 elements, the VL register is 7 bits wide, and the P register is 24 bits wide, which enables a program range of 4 Mwords. In Y-MP mode, only instructions defined in previous CRAY Y-MP systems are available.

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.

BIBLIOGRAPHY



BIBLIOGRAPHY

Related Cray Research, Inc. hardware manuals are listed below. Refer to Section 6 for a list of related software manuals.

Cray Research Peripheral Equipment Site Planning Reference Manual, CRI publication number HR-00080.

This manual provides site planning information about operator and maintenance workstation equipment, disk storage units (DSUs), and front-end interface (FEI) cabinets.

Cray Support Equipment Site Planning Reference Manual, CRI publication number HR-00082.

This manual provides site planning information about refrigeration condensing units (RCUs) and motor-generator sets (MGSs).

CRAY Y-MP C90 Site Planning Reference Manual, CRI publication number HR-04025.

This manual describes the physical requirements for the CRAY Y-MP C90 computer system. It defines customer and Cray Research, Inc. site planning and preparation responsibilities. This manual also describes the operational requirements, system configurations, mainframe and cooling unit specifications and requirements, and computer room floor specifications.

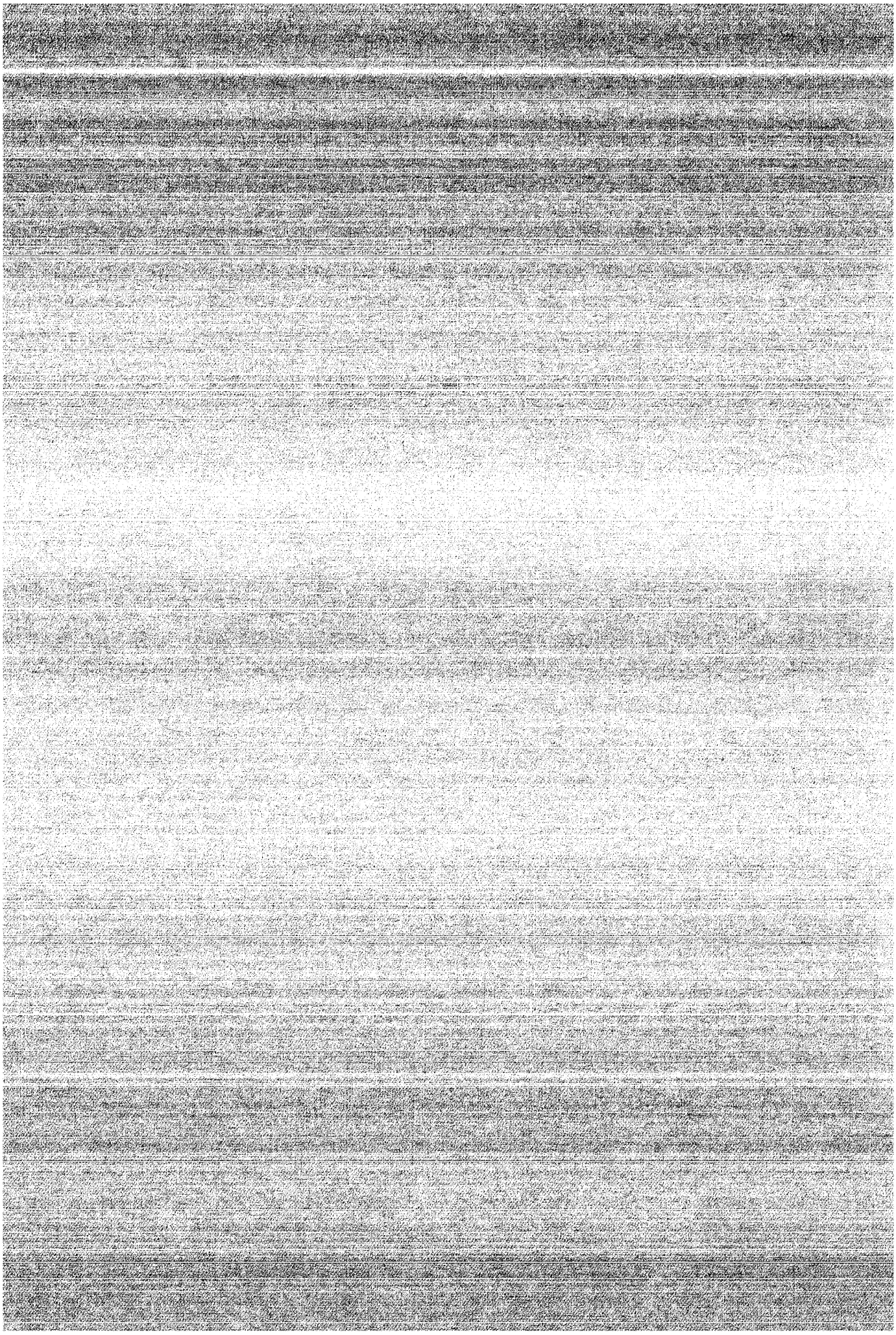
Principles of Computer Room Design, CRI publication number HR-04013.

This manual describes computer room design principles to help computer room facility managers prepare, inspect, and maintain a stable, problem-free environment. Information on computer room and raised-floor construction, system cooling, environmental control, fire and lightning protection, power, and grounding is also discussed.

Safe Use and Handling of Fluorinert Liquids, CRI publication number HR-0306.

This manual is written for Cray Research, Inc. customers and field engineers whose Cray Research computer system uses Fluorinert Liquid. The manual warns and informs about using Fluorinert Liquid and describes its uses at Cray Research, Inc. The manual describes the Material Safety Data Sheet and explains its significance in using Fluorinert Liquid or any other chemical.

INDEX



INDEX

Boldface numbers refer to illustrations and charts.

Reader Comment Form

Title: CRAY Y-MP C90 Functional Description Manual **Number: HR-04028**

Your feedback on this publication will help us provide better documentation in the future. Please take a moment to answer the few questions below.

For what purpose did you primarily use this manual?

_____Troubleshooting
_____Tutorial or introduction
_____Reference information
_____Classroom use
_____Other - please explain _____

Using a scale from 1 (poor) to 10 (excellent), please rate this manual on the following criteria and explain your ratings:

_____Accuracy _____
_____Organization _____
_____Readability _____
_____Physical qualities (binding, printing, page layout) _____
_____Amount of diagrams and photos _____
_____Quality of diagrams and photos _____

Completeness (Check one)

_____Too much information _____
_____Too little information _____
_____Just the right amount of information

Your comments help Hardware Publications and Training improve the quality and usefulness of your publications. Please use the space provided below to share your comments with us. When possible, please give specific page and paragraph references. We will respond to your comments in writing within 48 hours.

NAME _____
JOB TITLE_____
FIRM_____
ADDRESS_____
CITY_____STATE_____ZIP_____
DATE_____

[or attach your business card]

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.

Attn: Hardware Publications and Training
770 Industrial Boulevard
Chippewa Falls, WI 54729

STAPLE

Cray Research, Inc.
Hardware Publications and Training
770 Industrial Boulevard
Chippewa Falls, WI 54729