CRAY Y-MP® Computer Systems Functional Description Manual

HR-04001-0C

Cray Research, Inc.

Copyright © 1990 Cray Research, Inc. Portions of the Autotasking documentation Copyright © 1988 Pacific-Sierra Research Corporation. This manual or parts thereof may not be reproduced in any form unless permitted by contract or by written permission of Cray Research, Inc.

The CRAY Y-MP computer system is exempt from the technical requirements of the FCC's Part 15 Subpart J rules pursuant to Section 15.801 (C).

CRAY®, CRAY-1®, CRAY Y-MP®, HSX®, SSD®, and UNICOS® are federally registered trademarks and Autotasking™, CFT™, CFT77™, CFT2™, CRAY X-MP™, COS™, Cray Ada™, CRAY-2™, CRAY Y-MP2E™, CSIM™, Delivering the power . . .™, IOS™, OLNET™, RQS™, SEGLDR™, SUPERLINK™, and X-MP EA™ are trademarks of Cray Research, Inc.

Amdahl is a registered trademark of Amdahl Corporation. AOS is a registered trademark of Data General Corporation. Apollo and DOMAIN are registered trademarks of Apollo Computer Inc. CDC is a registered trademark of Control Data Corporation. ECLIPSE is a registered trademark of Data General Corporation. Ethernet is a registered trademark of the Xerox Corporation. Fluorinert liquid is a registered trademark of 3M. HYPERbus and HYPERchannel are registered trademarks of Network Systems Corporation. IBM is a registered trademark of International Business Machines Corporation. UNISYS is a registered trademark of UNISYS Corporation. UNIX is a registered trademark of AT&T.

AEGIS is a trademark of Apollo Computer Inc. CYBER is a trademark of Control Data Corporation. DEC, PDP, VAX, VAXcluster, and VMS are trademarks of Digital Equipment Corporation. Honeywell is a trademark of Honeywell, Incorporated. LANlord is a trademark of Computer Network Technology. Sun-3 is a trademark of Sun Microsystems, Inc. VMEbus is a trademark of Motorola, Inc. The UNICOS operating system is derived from the AT&T UNIX System V operating system. UNICOS is also based in part on the Fourth Berkeley Software Distribution under license from The Regents of the University of California.

MVS, VM, and VM/CMS are products of International Business Machines Corporation. NOS, NOS/BE, and NOS/VE are products of Control Data Corporation.

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 Computer Systems Functional Description Manual** **Number:** **HR-04001-0C**

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

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention: HARDWARE PUBLICATIONS
770 Industrial Boulevard
Chippewa Falls, WI 54729

Fold

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 a new alphabetic level.

Every page of a manual changed by a reprint with revision has the revision level indicated in the publication number in the ninth position. 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
	January 1988 - Original printing.
A	February 1989 - This revision adds information about new products: the CRAY Y-MP8, CRAY Y-MP4, and CRAY Y-MP2 models, and the DS-40D Disk Subsystem. This revision also incorporates readers' comments.
B	September 1989 - This revision adds register parity information for the CRAY Y-MP computer system. Change bars have been added to indicate where this register parity information was incorporated. Various technical and editorial changes have also been made.
B1	January 1990 - This change packet includes information on the new CRAY Y-MP 128-Mword central memory. All of the CRAY Y-MP specification sheets were updated and a new specification sheet for the FOL-3 fiber-optic link was added. The following changes were made: • On page vii, the following related publication was added: *Principles of Computer Room Design*, publication number HR-04013. • On page 1-6, the statement, "The MWS is not connected to the customer's network." was added. • On page 2-6, the corrected statement now reads, "Each CPU now contains 8 registers." • On page 2-37: "A 2-parcel or 3-parcel instruction begins in any parcel of a word and can span a word boundary." Change bars are included for your convenience.
C	June 1990. This revision incorporates change packet HR-04001-0B1 and adds information on the new DS-41 Disk Subsystem.

PREFACE

This manual describes the basic functions of the CRAY Y-MP computer system currently manufactured by Cray Research, Inc.

AUDIENCE

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

ORGANIZATION

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

SECTION 1 - CRAY Y-MP COMPUTER SYSTEM OVERVIEW introduces and describes the CRAY Y-MP system components and support equipment.

SECTION 2 - CRAY Y-MP MAINFRAME describes the basic architecture of the CRAY Y-MP mainframe. This section is divided into two subsections. The first subsection describes the hardware architecture of the mainframe. The second subsection describes the CPU instructions. Three specification sheets (one each for the CRAY Y-MP8, CRAY Y-MP4, and CRAY Y-MP2 computer systems) are included at the end of this section

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

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

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

SECTION 6 SOFTWARE OVERVIEW provides an overview of the software available for the CRAY Y-MP computer system.

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

NOTATIONAL CONVENTIONS

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.
Number base	All numbers used in this manual are decimal, unless otherwise indicated. Octal numbers are indicated with an 8 subscript. Exceptions are register numbers, the instruction parcel in instruction buffers, and instruction forms, which are given in octal without the subscript.

The following are examples of the preceding conventions.

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

RELATED PUBLICATIONS

For additional information on site planning, refer to the following publications.

- HR-00080 — The *Cray Peripheral Equipment Site Planning Reference Manual* provides site planning information for operator and maintenance workstation equipment, Disk Storage units (DSUs), and Front-end Interface (FEI) cabinets.

- HR-00082 — The *Cray Support Equipment Site Planning Reference Manual* provides site planning information for Refrigeration Condensing units (RCUs) and Motor-generator sets (MGSs).

- HR-00306 — The *Safe Use and Handling of Fluorinert Liquids* is written for Cray Research, Inc. customers and field engineers whose Cray computer system uses Fluorinert liquid, 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.

- HR-04000 — The *CRAY Y-MP8 Computer Systems Site Planning Reference Manual* provides site planning information for the CRAY Y-MP mainframe, the mainframe Heat Exchanger Unit (HEU), the I/O Subsystem (IOS), the SSD Solid-state Storage Device, and the IOS and SSD Power Distribution Units (PDUs).

- HR-04002 — The *CRAY Y-MP2 Computer Systems Site Planning Reference Manual* provides site planning information for the CRAY Y-MP2 computer system. It contains technical information to plan and prepare a typical site for installing a CRAY Y-MP2 computer system.

- HR-04003 — The *CRAY Y-MP4 Computer Systems Site Planning Reference Manual* provides site planning information for the CRAY Y-MP4 computer system. It contains technical information to plan and prepare a typical site for installing a CRAY Y-MP4 computer system.

- HR-04013 — The *Principles of Computer Room Design* manual describes computer room design principles to help computer room facility managers prepare, inspect, and maintain a stable, problem-free environment. Computer room and raised-floor construction, system cooling, environmental control, fire and lightning protection, power, and grounding are also discussed.

- SN-03030 — The *Operator Workstation (OWS) Guide* describes the commands and operation of the VME-based OWS used for CRAY Y-MP and CRAY X-MP EA computer system operation and monitoring. This manual is for computer operators and system administrators.

- SR-00085 The *Symbolic Machine Instructions Reference Manual* describes the machine instructions used on CRAY-1, CRAY X-MP, and CRAY Y-MP computer systems.

A list of related software publications is included at the end of Section 6, "Software Overview."

Please use one of the reader comment forms located at the front and back of this manual to suggest improvements or point out technical errors.

CONTENTS

1 - CRAY Y-MP COMPUTER SYSTEM OVERVIEW 1-1
- CRAY Y-MP Mainframe 1-3
- I/O Subsystem 1-3
- SSD Solid-state Storage Device 1-4
- Disk Storage Units 1-5
- Network Interfaces 1-5
- Operator and Maintenance Workstations 1-6
- Power and Cooling Support Equipment 1-6

2 - CRAY Y-MP MAINFRAME 2-1
- CPU Shared Resources 2-1
- CPU Computation Section 2-3
- CPU Control Section 2-22
- Special Features of the CRAY Y-MP Computer System 2-29
- CPU INSTRUCTIONS 2-37
- CRAY Y-MP8 Computer System Specification Sheet 2-67
- CRAY Y-MP4 Computer System Specification Sheet 2-71
- CRAY Y-MP2 Computer System Specification Sheet 2-75

3 - I/O SUBSYSTEM 3-1
- I/O Processors 3-1
- I/O Subsystem Buffer Memory 3-4
- System Operator Workstation 3-4
- I/O Subsystem Model D Specification Sheet 3-5

4 - SSD SOLID-STATE STORAGE DEVICE 4-1
- SSD Functions 4-1
- SSD Memory Size 4-2
- SSD Memory Transfer and Data Protection 4-2
- SSD Solid-state Storage Device Specification Sheet 4-3

5 - PERIPHERAL EQUIPMENT 5-1
- Disk Controller Units and Disk Storage Units 5-1
  - Disk Controller Units 5-1
  - Disk Storage Units 5-1
    - DS-40 Disk Subsystem 5-2
      - DS-40 Disk Subsystem Standard Configurations 5-3
      - DS-40D Disk Daisy Chain Configurations 5-3

PERIPHERAL EQUIPMENT (continued)

DS-41 Disk Subsystem 5-3
DS-41 Disk Subsystem Standard Configurations 5-4
DS-41A Disk Subsystem Field-upgradable Configurations 5-4
DS-41D Disk Daisy Chain Configurations 5-4
DS-41R Disk Subsystem Redundant Configurations 5-5
DD-49 Disk Storage Unit 5-5
Network Interfaces 5-6
FEI-1 Front-end Interface 5-6
Fiber-optic Link 5-6
FEI-3 Front-end Interface 5-7
Direct Network Connections 5-7
High-speed External (HSX) Communications Channel 5-7
High Performance Parallel Interface (HiPPI) 5-7
DEC VAX Supercomputer Gateway 5-8
DS-40 and DS-40D Disk Subsystems Specification Sheet 5-9
DS-41, DS-41D, and DS-41R Disk Subsystems Specification Sheet 5-11
DD-49 Disk Drive Specification Sheet 5-13
Front-end Interface Specification Sheet 5-15
FOL-3 Fiber-optic Link Specification Sheet 5-17

6 - SOFTWARE OVERVIEW 6-1
Operating Systems 6-1
Multiprocessing 6-2
Fortran Compilers 6-3
C Compiler 6-4
Pascal 6-4
Cray Assembler 6-5
Subroutine Libraries 6-5
Utilities 6-5
I/O Subsystem Software 6-6
Communications Software 6-6
Applications 6-7
Software Publications 6-7
Software Training 6-10

GLOSSARY GL-1

INDEX Index-1

CONTENTS

1 - CRAY Y-MP COMPUTER SYSTEM OVERVIEW 1-1

CRAY Y-MP Mainframe 1-3

I/O Subsystem 1-3

SSD Solid-state Storage Device 1-4

Disk Storage Units 1-5

Network Interfaces 1-5

Operator and Maintenance Workstations 1-6

Power and Cooling Support Equipment 1-6

FIGURES

Figure 1-1. CRAY Y-MP8 Computer System 1-2

Figure 1-2. I/O Subsystem Chassis 1-3

Figure 1-3. SSD Solid-state Storage Device Chassis 1-4

Figure 1-4. Typical Front-end Interface Cabinet 1-5

Figure 1-5. CRAY Y-MP Mainframe Cooling System 1-7

Figure 1-6. Refrigeration Condensing Unit 1-8

Figure 1-7. Motor-generator Cabinet 1-9

Figure 1-8. IOS and SSD Power Distribution Unit 1-9

1 - CRAY Y-MP COMPUTER SYSTEM OVERVIEW

The CRAY Y-MP computer system is a powerful, general-purpose supercomputer. The large memory and fast clock speed of the CRAY Y-MP computer system allow for faster throughput, allowing for more efficient use of computing power. The CRAY Y-MP computer system is able to achieve extremely high multiprocessing rates by efficiently using the scalar and vector processing capabilities of the multiple Central Processing Units (CPUs), combined with the systems' solid-state, random access memory (RAM), and shared registers.

The CRAY Y-MP series consists of three models: CRAY Y-MP8, CRAY Y-MP4, and CRAY Y-MP2 computer systems. The official naming convention for the CRAY Y-MP series is CRAY Y-MPn/xy, where n, x, and y represent the following numbers:

- n = maximum number of CPUs the mainframe can house
- x = number of processors in a particular configuration
- y = number of Mwords of central memory in a particular configuration

The chassis are not field upgradable beyond their maximum CPU configuration. For specific information concerning CPU and memory configurations, refer to the specification sheets at the end of Section 2.

The CRAY Y-MP computer system is carefully balanced to deliver optimum overall performance. The unique architecture of CRAY Y-MP computer systems allows faster and more efficient use of the vector and scalar processing capabilities inherent in all Cray computer systems.

Vector processing uses a single instruction to perform multiple operations on sets of ordered data. 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 on the CRAY Y-MP computer system is short enough so that vector processing is more efficient than scalar processing for vectors containing as few as two elements. This feature allows for fast long and short vector processing to be balanced with high-speed scalar processing, while both are supported by powerful input/output capabilities.

Multiple-processor CRAY Y-MP computer 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.

The CRAY Y-MP computer system consists of a mainframe, one or two I/O Subsystems (IOSs), and an optional SSD Solid-state Storage Device (SSD). Figure 1-1 shows a typical CRAY Y-MP computer system. Mass storage devices (such as disk and tape drives) and front-end interfaces (FEIs) can also be configured with the system.

Support equipment for the mainframe includes a Heat Exchanger Unit (HEU) and Refrigeration Condensing Unit (RCU) for cooling. The Power Distribution Unit (PDU) for the mainframe is located inside the mainframe; 400-Hz power is supplied by the mainframe's Motor-generator Set (MGS). Support equipment for the IOS and SSD include RCUs, PDUs, and an MGS. The following subsections introduce the system components; later sections provide more detailed information on the IOS, SSD, FEIs, and mass storage devices.

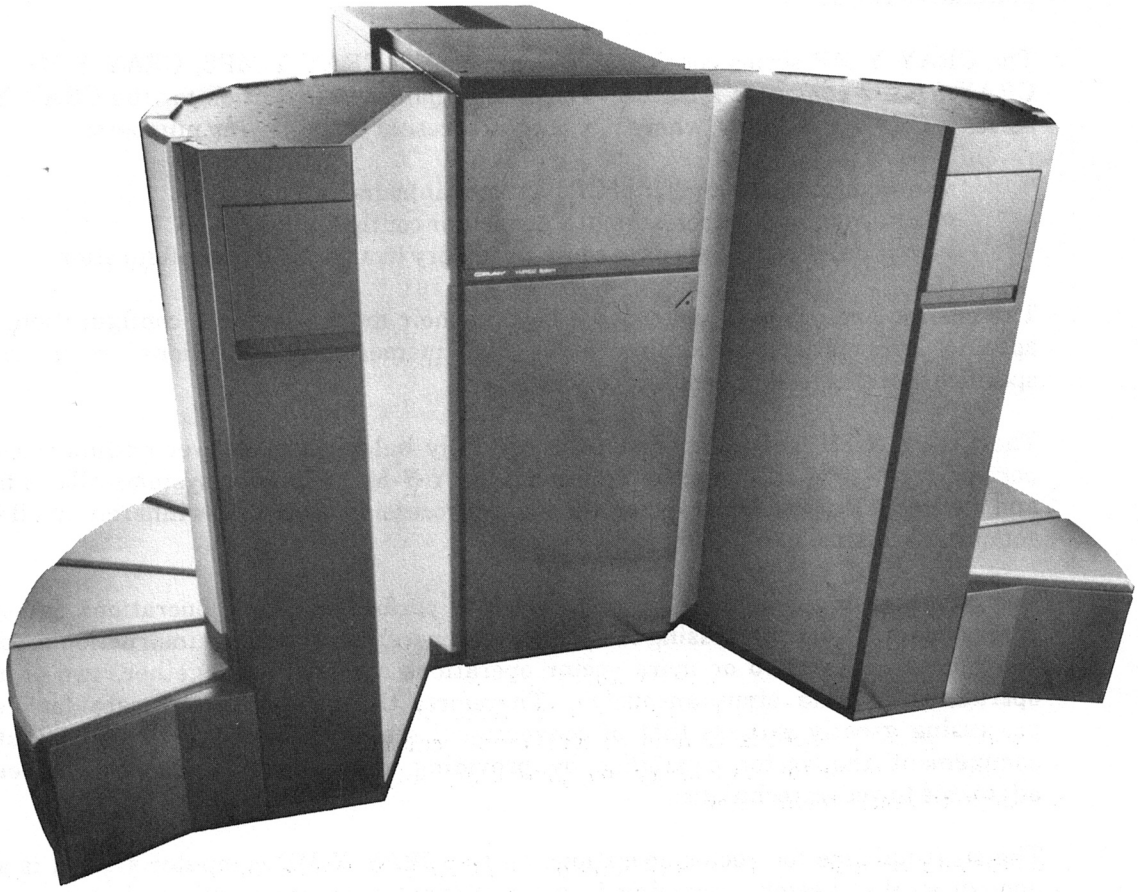


Figure 1-1. CRAY Y-MP Computer System

CRAY Y-MP MAINFRAME

The CRAY Y-MP mainframe contains the Central Processing Units (CPUs), an I/O section, an Interprocessor Communication section, a Real-time Clock (RTC), and Central Memory. The I/O section, Interprocessor Communication section, Real-time Clock, and Central Memory are shared by all CPUs in multiprocessor computer systems.

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

Refer to Section 2 for more specific information on the CRAY Y-MP mainframes.

I/O SUBSYSTEM

The CRAY Y-MP computer system includes an I/O Subsystem (IOS); a second IOS is optional with the CRAY Y-MP8 computer system. Each IOS (a single IOS chassis, referred to as the IOC, is shown in Figure 1-2) has multiple I/O Processors (IOPs), a Buffer Memory, and required interfaces. It is designed for fast data transfer between front-end computers, peripheral devices, storage devices, and the IOS's Buffer Memory, or between its Buffer Memory and the Central Memory of the CRAY Y-MP mainframe.

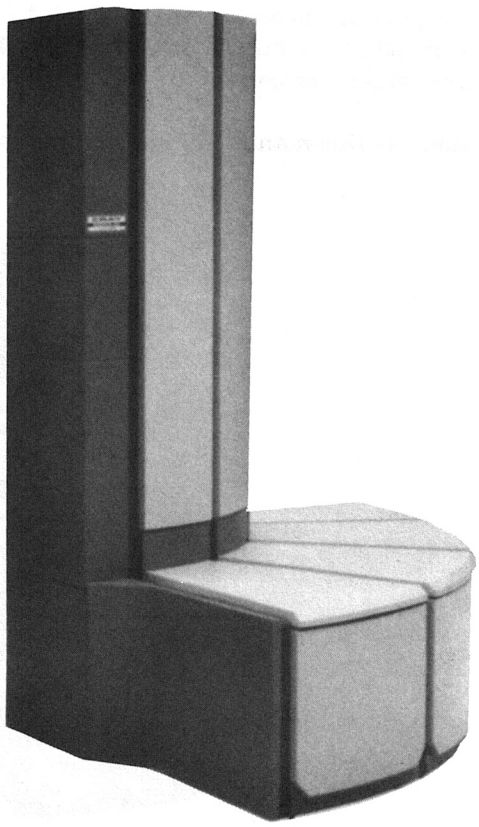


Figure 1-2. I/O Subsystem Chassis

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

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

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

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

SSD SOLID-STATE STORAGE DEVICE

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

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

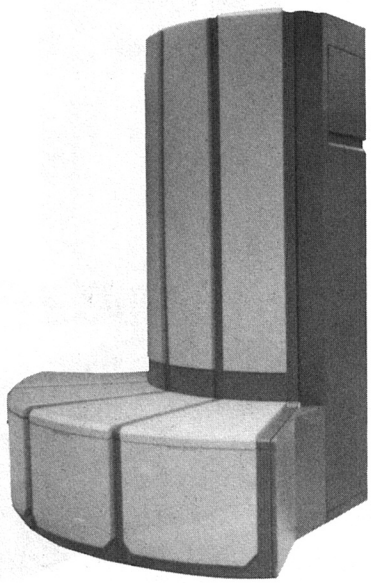


Figure 1-3. SSD Solid-state Storage Device Chassis

DISK STORAGE UNITS

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

NETWORK INTERFACES

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

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

Some FEI's are housed in a stand-alone cabinet located near the host computer (refer to Figure 1-4), while some install directly into the front-end computer system. In either case, operation of the FEI is invisible to both the front-end and Cray user.

As an option, a fiber-optic link is available for some FEIs to provide front-end connections of up to 3,280 ft (1,000 m) and complete electrical separation from the CRAY Y-MP computer system.

The CRAY Y-MP mainframe can be connected to computer networks directly, or through a front-end computer system. Refer to "Network Interfaces" in Section 5 of this manual for specific information.

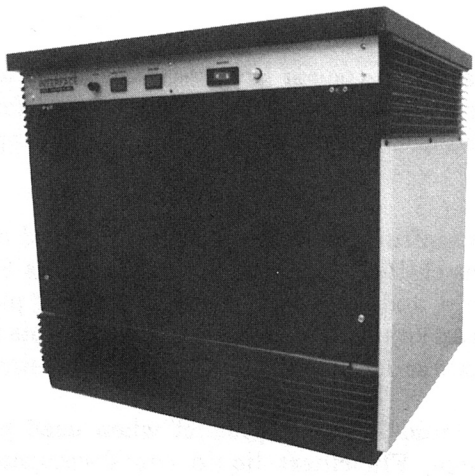


Figure 1-4. Typical Front-end Interface Cabinet

OPERATOR AND MAINTENANCE WORKSTATIONS

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

- System operator interface
- System deadstart and master clear functions
- Software maintenance utilities
- Local tape and local printer control
- System time-of-day clock

In addition, the OWS provides enhanced features, such as a Control Subsystem Network interface, which can be used to network workstations in a multiple system site or for multiple system operators.

The OWS communicates with the CRAY Y-MP computer system through a 6-Mbyte/s I/O channel from an IOP in the IOS. The tape drives, disks, printer, and time-of-day clock are available to the mainframe over this 6-Mbyte/s channel.

The MWS is a microcomputer system used for hardware maintenance and monitoring. The MWS is owned by Cray Research, Inc. and is supplied as part of the maintenance contract and therefore is not part of the customer's system. The MWS is not connected to the customer's network.

POWER AND COOLING SUPPORT EQUIPMENT

CRAY Y-MP computer systems require support equipment for power and cooling. Power is supplied by MGSs and PDUs. The system cooling components include a Heat Exchanger Unit (HEU) and RCUs. The remainder of this section defines and explains the various mainframe, IOS, and SSD support equipment. Refer to the appropriate Site Engineering manuals listed in the Preface for more information on power and cooling requirements.

The CRAY Y-MP mainframe power supplies and voltage-adjusting controls are located in the mainframe chassis; a separate PDU is not needed for the mainframe. The 400-Hz power from the MGSs is distributed among the power supplies (MGSs are described later in this section).

The CRAY Y-MP mainframe is cooled by an HEU and an RCU. The HEU contains a pump that circulates chilled dielectric coolant (such as Fluorinert liquid) through each module, power supply, and the power supply mounting plate (refer to Figure 1-5). Each circulation loop has an adjustable ball valve that controls the flow rate. The temperature of the modules and power supplies is changed by increasing or decreasing the flow rate.

Note: Fluorinert liquid is a safe product when used properly. When exposed to an excessive heat source, Fluorinert liquid can decompose and produce hazardous by-products. The *Safe Use and Handling of Fluorinert Liquids* Manual (publication number HR-0306) provides specific guidelines and information regarding Fluorinert liquid.

The mainframe RCU cools the dielectric coolant in the HEU with a refrigerant. The RCU itself is cooled by customer-supplied chilled water; the RCU is described later in this section.

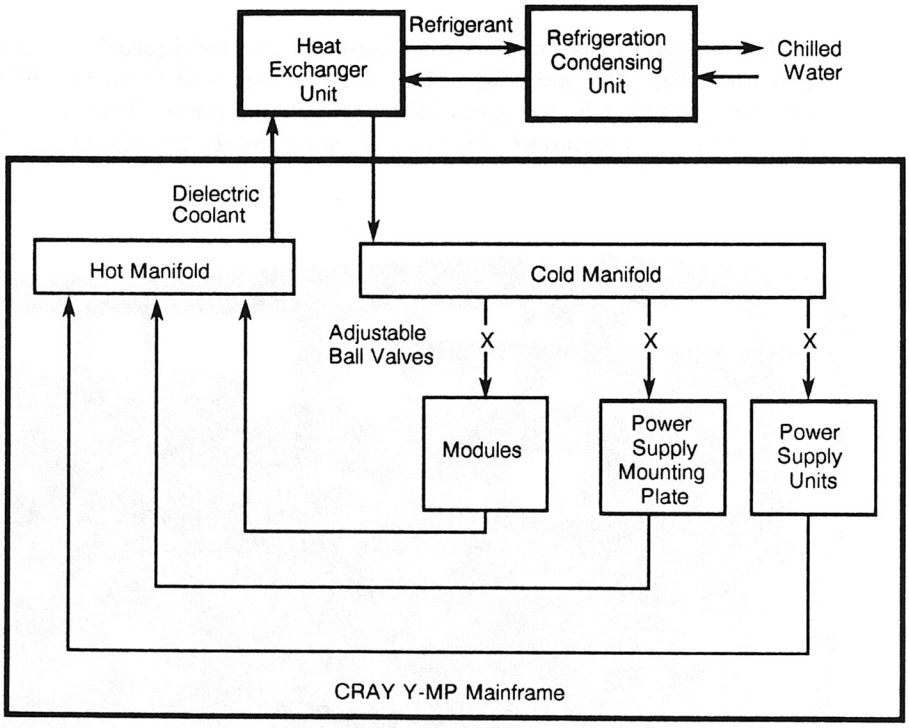


Figure 1-5. CRAY Y-MP Mainframe Cooling System

Because of the intense heat created by the density of the circuitry, the CRAY Y-MP mainframe has a monitoring system consisting of PC boards and a system control panel. This panel is located at the power supply end of the mainframe chassis. The panel contains indicators that allow the following functions to be monitored:

- DC voltages - provides voltage level monitoring, loss of voltage protection, and over-voltage protection.
- AC voltages - monitors MGS phase-to-phase and phase-to-neutral voltages.
- Temperatures - monitors power supply mounting plate 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 module plate flow rates to protect the mainframe from overheating.

The monitoring system provides a method for the system attendant to observe these conditions. The monitoring system also functions as a backup system that can automatically shut down the mainframe if an abnormal condition exists and the system attendant fails to notice the condition. In addition to these automatic monitoring features, the AC output voltage on the MGSs can be set from this panel.

An RCU contains the major components of the refrigeration system used to cool the mainframe, the IOS, and the SSD. Heat is removed from the RCU by a second-level cooling system that is not part of the computer system. The number of RCUs needed is system- and site-dependent. Figure 1-6 shows the RCU without the covers.

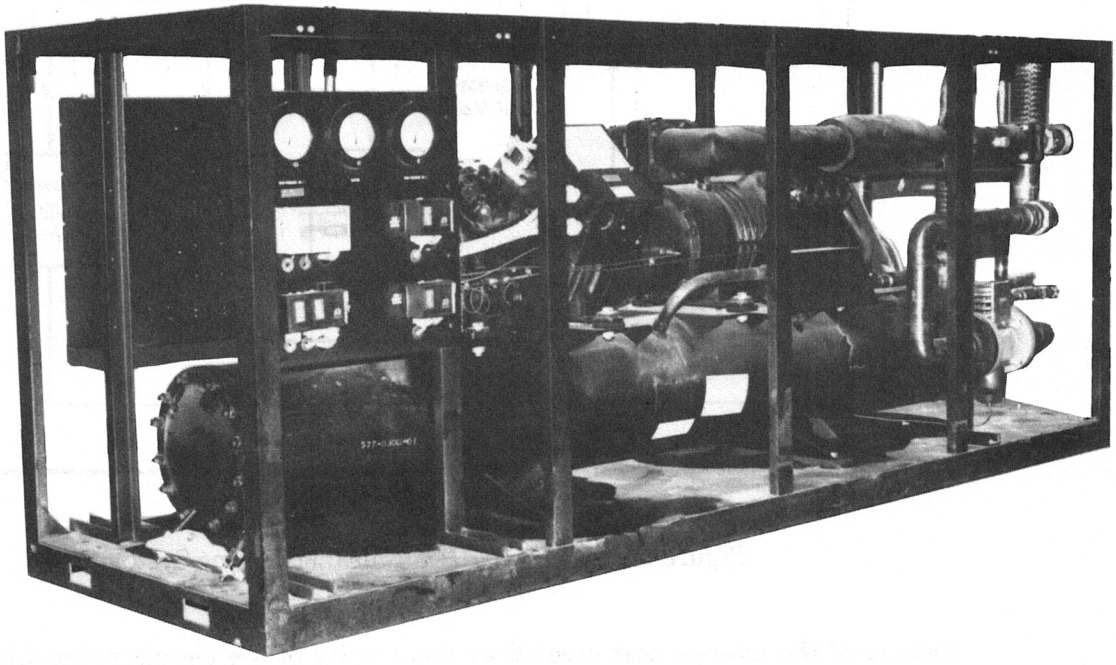


Figure 1-6. Refrigeration Condensing Unit

The CRAY Y-MP computer system contains up to three MGSs. The maximum configuration includes one MGS for the mainframe, one MGS for the IOS and SSD, and a standby. Each MGS converts primary power from commercial power mains to the 400-Hz power used by the mainframe, the IOS, and the SSD. The MGS also isolates these components from transients and fluctuations on the commercial power mains. An MGS cabinet is shown in Figure 1-7.

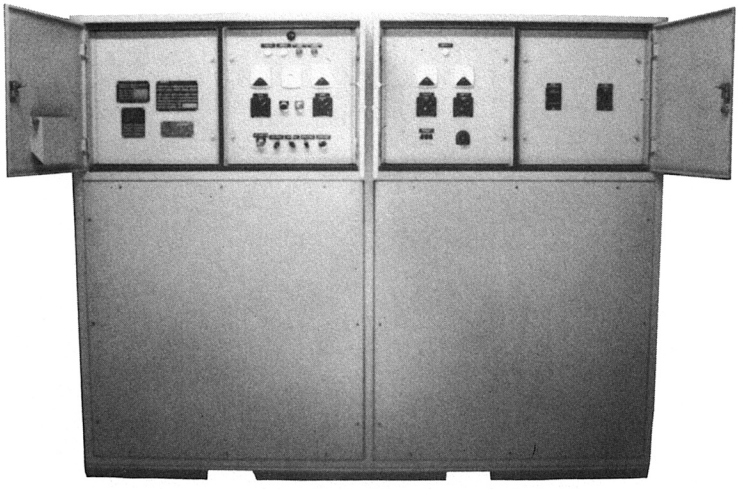


Figure 1-7. Motor-generator Cabinet

The IOS and SSD each have independent PDUs. The PDU contains temperature and voltage-monitoring equipment that checks temperatures at strategic locations on the IOS and SSD chassis (the automatic warning system alerts the system attendant if overheating or excessive cooling occurs). If the system attendant fails to notice these conditions, the PDU powers down the equipment. Figure 1-8 shows the PDU used by the IOS and SSD.

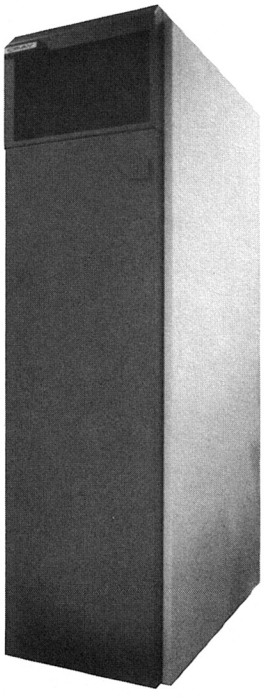


Figure 1-8. IOS and SSD Power Distribution Unit

CONTENTS

2 - CRAY Y-MP MAINFRAME 2-1
CPU Shared Resources 2-1
Central Memory 2-1
I/O Section 2-2
Interprocessor Communication Section 2-2
Real-time Clock 2-2
CPU Computation Section 2-3
Registers 2-5
Address Registers 2-5
Scalar Registers 2-6
Vector Registers 2-6
Functional Units 2-7
Address Functional Units 2-7
Scalar Functional Units 2-7
Vector Functional Units 2-8
Floating-point Functional Units 2-9
Functional Unit Operations 2-10
Logical Operations 2-10
Integer Arithmetic 2-11
Floating-point Arithmetic 2-12
CPU Control Section 2-22
Exchange Mechanism 2-22
Exchange Sequence 2-22
Exchange Package 2-22
Instruction Fetch 2-27
Instruction Issue 2-27
Programmable Clock 2-28
Performance Monitor 2-28
Status Register 2-28
Special Features of the CRAY Y-MP Computer System 2-29
Pipelining and Segmentation 2-29
Functional Unit Independence 2-29
Vector Processing 2-30
Definition of Vector Processing 2-30
Advantages of Vector Processing 2-31
Vector Chaining 2-31
Types of Vector Instructions 2-32

CPU Instructions 2-37
Instruction Formats 2-37
1-parcel Instruction Format with Discrete *j* and *k* Fields 2-37
1-parcel Instruction Format with Combined *j* and *k* Fields 2-38
2-parcel Instruction Format with Combined *j*, *k*, and *m* Fields 2-38
2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields 2-39
3-parcel Instruction Format with Combined *m* and *n* Fields 2-40
Instruction Differences Between X-Mode and Y-Mode 2-40
Special Register Values 2-42
Monitor Mode Instructions 2-43
Special CAL Syntax Forms 2-43
CPU Instruction Summary 2-43
Functional Units Instruction Summary 2-44
Functional Instruction Summary 2-45
CRAY Y-MP8 Computer System Specification Sheet 2-67
CRAY Y-MP4 Computer System Specification Sheet 2-71
CRAY Y-MP2 Computer System Specification Sheet 2-75

FIGURES

Figure 2-1. 8-processor CRAY Y-MP Computer System Block Diagram 2-4
Figure 2-2. Integer Data Formats 2-11
Figure 2-3. 24-bit Integer Multiply Performed in Floating-point Multiply Functional Unit 2-12
Figure 2-4. 32-bit Integer Multiply Performed in Floating-point Multiply Functional Unit 2-13
Figure 2-5 Floating-point Data Format 2-13
Figure 2-6. Internal Representation of Floating-point Number 2-14
Figure 2-7 Biased and Unbiased Exponent Range 2-15
Figure 2-8. Floating-point Add and Multiply Range Errors 2-16
Figure 2-9. Floating-point Reciprocal Approximation Range Errors 2-16
Figure 2-10. Newton's Method of Approximation 2-19
Figure 2-11. Segmentation and Pipelining Example 2-30
Figure 2-12 Vector Chaining Example 2-32
Figure 2-13 Vector vector Operand Instructions 2-33
Figure 2-14. Vector-scalar Operand Instructions 2-34
Figure 2-15. Vector Memory Instructions 2-34
Figure 2-16. Gather Instruction Example 2-35
Figure 2-17. Scatter Instruction Example 2-36
Figure 2-18. Compressed Index Example 2-36
Figure 2-19. General Format for Instructions 2-37
Figure 2-20. 1-parcel Instruction Format with Discrete *j* and *k* Fields 2-38
Figure 2-21. 1-parcel Instruction Format with Combined *j* and *k* Fields 2-38
Figure 2-22. 2-parcel Instruction Format with Combined *j*, *k*, and *m* Fields 2-39
Figure 2-23. 2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields 2-39

FIGURES (continued)

Figure 2-24. 2-parcel Instruction Format for a 24-bit Immediate Constant with Combined *i*, *j*, *k*, and *m* Fields 2-40
Figure 2-25. 3-parcel Instruction Format with Combined *m* and *n* Fields 2-41

TABLES

Table 2-1. CRAY Y-MP 3-parcel Instruction 2-41
Table 2-2. CRAY Y-MP/X-MP Instruction Differences 2-42
Table 2-3. Special Register Values 2-42

2 - CRAY Y-MP MAINFRAME

This section describes the major functional areas of a CRAY Y-MP mainframe, special features of the mainframe, and a summary of the Cray Assembly Language (CAL) instruction set. For specific information concerning the CRAY Y-MP8, CRAY Y-MP4, and CRAY Y-MP2 systems, refer to the specification sheets at the end of this section.

CPU SHARED RESOURCES

The Central Processing Units (CPUs) of the CRAY Y-MP computer system share several functional areas (or sections) of the mainframe. These sections include Central Memory, the I/O section, the Interprocessor communication section, and the Real-time Clock. The following subsections describe these functional areas.

Central Memory

The CRAY Y-MP Central Memory is shared by the CPUs and the I/O section. Central Memory is divided into interleaved banks. This arrangement improves memory access speed by allowing simultaneous and overlapping memory references. Simultaneous references are two or more references that begin at the same time. Overlapping references are one or more references that begin while another reference is in progress. Refer to the specification sheets at the end of this section for more information on memory size and number of banks for each model.

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

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

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

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

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

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

I/O Section

The I/O section is shared by all CPUs in multiprocessor computer systems. The mainframe supports three channel types identified by their maximum transfer rates: 6 Mbyte/s, 100 Mbyte/s, and 1000 Mbyte/s. The 6-Mbyte/s channels are used to transfer control information between the mainframe and a Cray I/O Subsystem (IOS). The 100-Mbyte/s channels are used to transmit data between the mainframe and an IOS. The 1000-Mbyte/s channels transfer data between the mainframe and an SSD solid-state storage device (SSD). The IOS and SSD are high-speed data transfer devices designed to support CRAY Y-MP mainframe processing. Refer to the specification sheets at the end of this section for more information on channel configurations for the different models.

Interprocessor Communication Section

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

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

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

Real-time Clock

The CRAY Y-MP mainframe contains one Real-time Clock (RTC) that is shared by all the CPUs. This clock consists of a 64-bit counter that advances one count each clock period (CP). Because the clock advances synchronously with program execution, it can be

used to time the program to an exact number of CPs. Contents of the RTC register can be read into or loaded from a Scalar (S) register.

CPU COMPUTATION SECTION

Each CPU is an identical, independent computation section consisting of operating registers, functional units, and an instruction control network. Refer to Figure 2-1 which shows the computation section of CPU 1 for an 8-processor CRAY Y-MP8 computer system. The operating registers and functional units of each computation section are associated with three types of processing: address, scalar, and vector.

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

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

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

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

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

Data flow in a 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. Data flows along either the scalar or vector path, depending on the processing mode. An exception is that scalar registers can provide one required operand for some vector instructions.

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

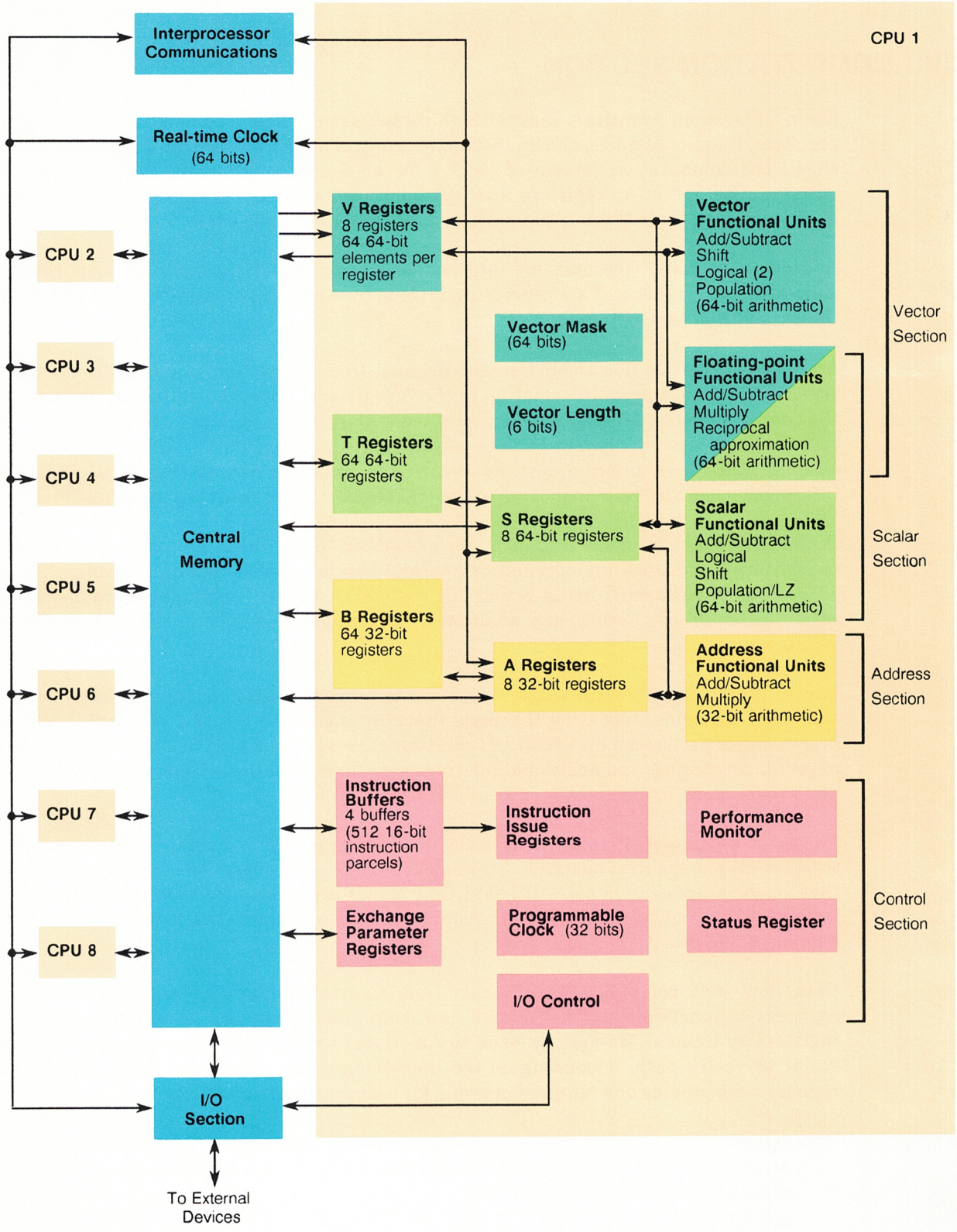


Figure 2-1. 8-processor CRAY Y-MP Computer System Block Diagram

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

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

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

The integer product is a scalar instruction designed for index calculation. Full indexing capability is possible throughout memory in either scalar or vector 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.

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

Registers

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

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

Address Registers

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

Each CPU contains 64 B registers; each register is 32 bits wide. The B registers are used as intermediate storage for the A registers. Data is transferred between B registers and Central Memory, and between A and B registers. Typically, B registers contain data to be referenced repeatedly over a long time span, making it unnecessary to retain the data

in either A registers or Central Memory. Examples of data stored in B registers are loop counts, variable array base addresses, and dimensions.

The 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 the word is read out of the B register. An error is indicated when the two sets do not match.

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

Scalar Registers

Each CPU contains 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 for scalar arithmetic and logical instructions. Scalar registers can also provide an operand for some vector operations.

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

The 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 the word is read out of the T register. An error is indicated when the two sets do not match.

Vector Registers

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

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

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

The V registers are 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.

Refer to "Vector Processing" later in this section for more information on vector processing.

Functional Units

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

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

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

Address Functional Units

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

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

Scalar Functional Units

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

The Scalar Add, Scalar Shift, Scalar Logical, and Scalar Population/Parity/Leading Zero functional units are used exclusively with scalar operations and are described here.

Three additional functional units are used for both scalar and vector operations. They are described in the following "Floating-point Functional Units" subsection. The following list describes the four Scalar functional units.

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

- The Scalar Shift functional unit shifts the entire contents of an S register or shifts the contents of two concatenated S registers 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 Logical functional unit performs bit-by-bit manipulation of quantities obtained from S registers.

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

Vector Functional Units

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

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

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

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

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

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

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

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

Floating-point Functional Units

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

- The Floating-point Add functional unit performs addition or subtraction of operands in floating-point format. The final result is normalized even when operands are unnormalized. (Refer to "Normalized Floating-point Numbers" later in this section for more information on normalized numbers.) Out-of-range exponents are detected.

- The Floating-point Multiply functional unit executes instructions that provide for 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 are assumed to be normalized. The Floating-point Multiply functional unit delivers a normalized result only if both input operands are normalized.

  Out-of-range exponents are detected. If both operands have zero exponents, however, the result is considered as 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 is assumed to be

normalized. The high-order bit of the coefficient is not tested, but is assumed to be a 1. Out-of-range exponents are detected.

Functional Unit Operations

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

Logical Operations

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

A logical product is the AND function; which is shown below.

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

A logical sum is the inclusive OR function; which is shown below.

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

A logical difference is the exclusive OR function; which is shown below.

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

A logical equivalence is the exclusive NOR function; which is shown below.

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

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

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

Integer Arithmetic

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

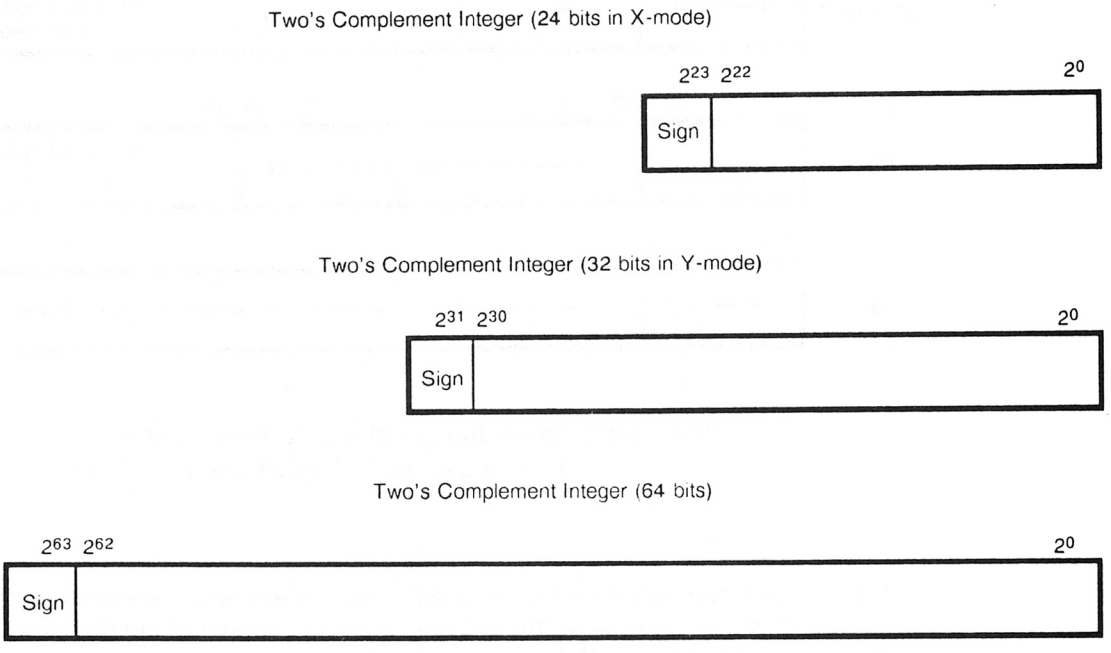


Figure 2-2. Integer Data Formats

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

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

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

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

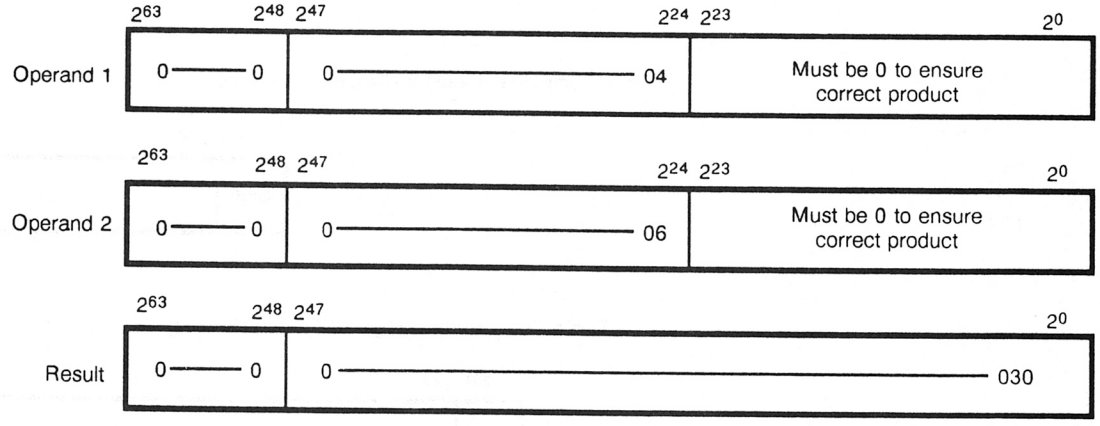


Figure 2-3. 24-bit Integer Multiply Performed in Floating-point Multiply Functional Unit

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

A second integer multiply operation performs a 32-bit multiply of the contents of S*j* and the contents of V*k* to V*i*. The operands must be left-shifted before the operation begins. The operand contained in S*j* must be left-shifted 31_{10} places, leaving the operand in bit positions 2^{62} through 2^{31}; bit positions 2^{30} through 2^0 must be equal to 0 to ensure accuracy (refer to Figure 2-4). The operand contained in V*k* must be left-shifted 16_{10} places, leaving the operand in bit positions 2^{47} through 2^{16}; bit positions 2^{15} through 2^0 must be equal to 0 to ensure accuracy. The result of the multiply is right-justified into positions 2^{31} through 2^0, while positions 2^{32} through 2^{63} are zero-filled.

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

Floating-point Arithmetic

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

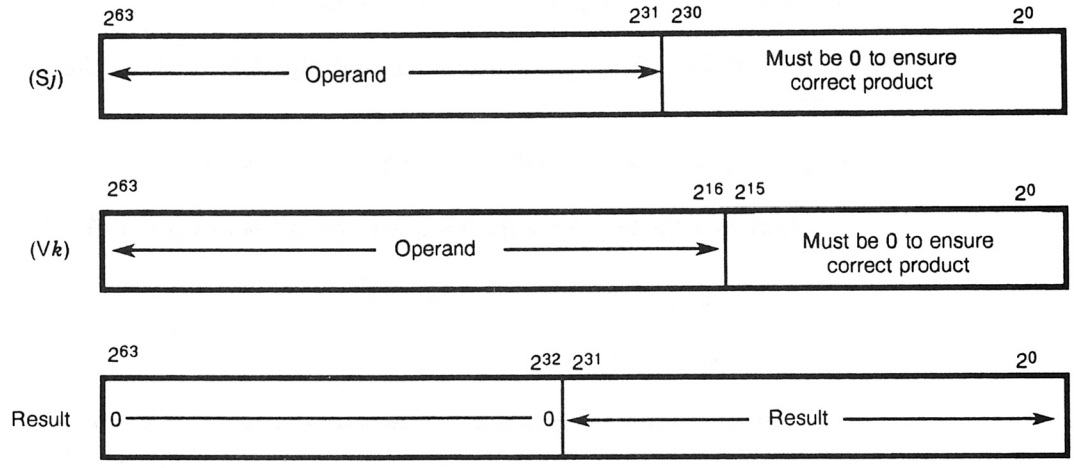


Figure 2-4. 32-bit Integer Multiply Performed in Floating-point Multiply Functional Unit

Floating-point Data Format

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

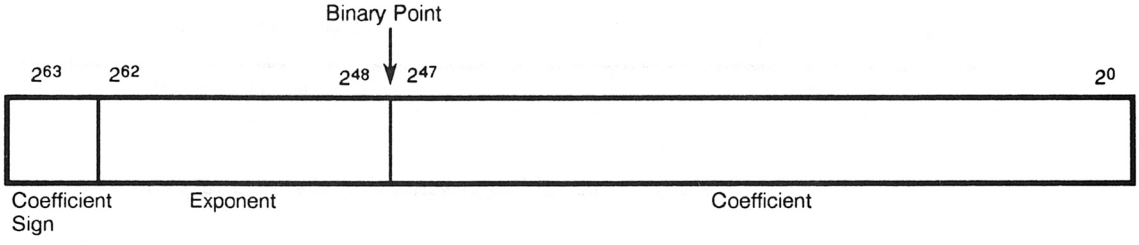


Figure 2-5. Floating-point Data Format

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

The coefficient is a 48-bit signed fraction; the sign of the coefficient is located in bit position 2^{63}. Because the coefficient is in sign-magnitude format, it is not complemented

for negative values. A normalized floating-point number has a 1 in the 2^{47} bit position, while an unnormalized floating-point number has a 0 in this bit position (normalized numbers are discussed in more detail later in this section).

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

To convert a floating-point number to its decimal equivalent:

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 0s. 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.

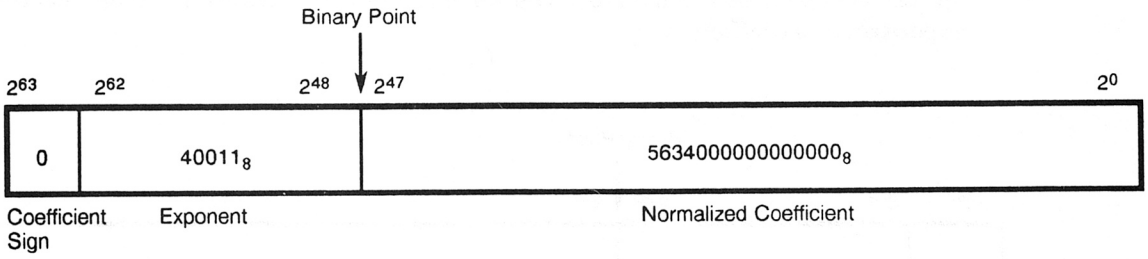


Figure 2-6. Internal Representation of Floating-point Number

Exponent Ranges

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

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

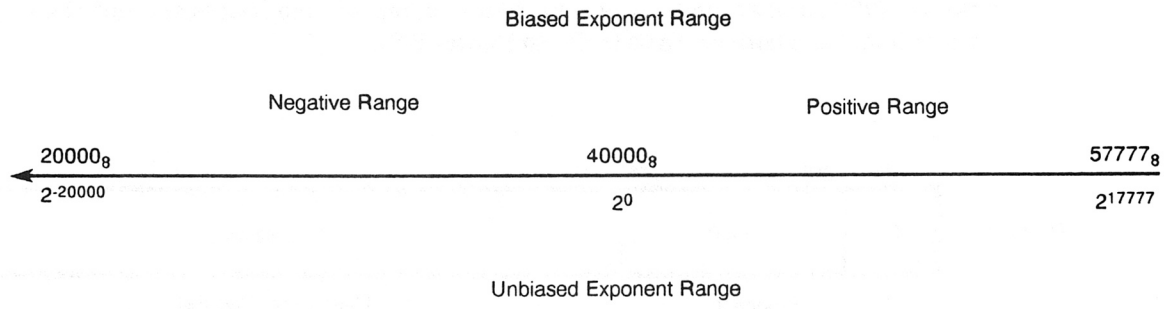


Figure 2-7. Biased and Unbiased Exponent Range

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.

When a floating-point number is created by inserting an exponent of 40060_8 and a 48-bit integer word into the coefficient, the result should be normalized before being used in a floating-point operation. Normalization is accomplished by adding the unnormalized floating-point operand to 0.

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

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

Floating-point Range Errors

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

When an overflow condition is detected, an interrupt occurs only if the Interrupt-on floating-point Error (IFP) bit is set in the Mode register and the system is not in monitor mode. The IFP flag can be set or cleared by a user mode program.

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

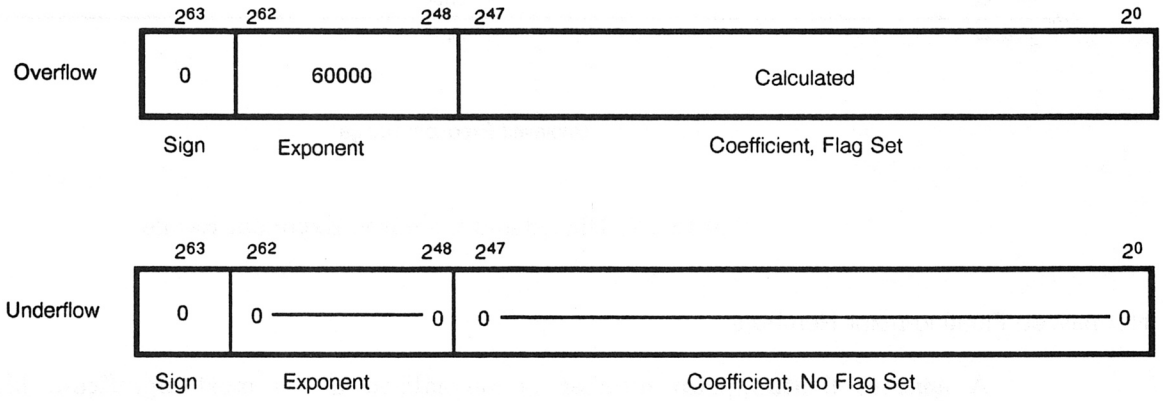


Figure 2-8. Floating-point Add and Multiply Range Errors

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

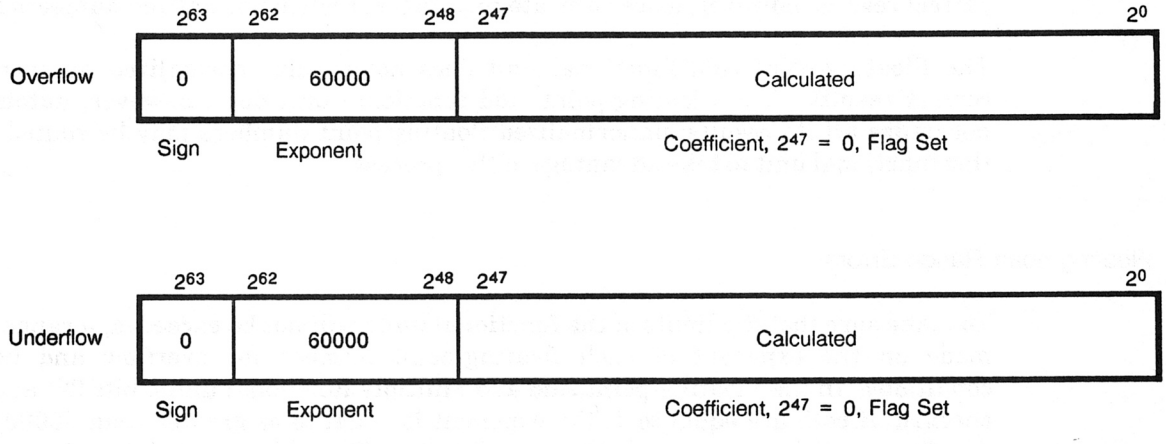


Figure 2-9. Floating-point Reciprocal Approximation Range Errors

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

Floating-point Addition Algorithm

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

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

After equalizing the two coefficients, 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 add 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 subtract operation is performed.
- If the sign bits are not equal and a subtract instruction is issued, an add operation is performed.

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

The normalization feature of the Floating-point Add functional unit can be used to normalize any floating-point number. Simply pair it with a zero operand and send them 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 section for more information on how the result is checked.

Floating-point Multiplication Algorithm

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

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

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

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

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

For a half-precision multiply, 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 up and the 29 most significant bits of the normalized result are transmitted back.

The variations due to this truncation and rounding are in the following range:

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

or

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

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

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

Floating-point Division Algorithm

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

Obtaining a quotient for two floating-point numbers involves two general steps. For example, to solve the equation 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 of A × 1/B.

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

To find the reciprocal, the equation, F(x) = 1/x - B, must be solved. To do this, a number, A, must be found so that F(A) = 1/A - B = 0. That is, the number A is the root of the equation 1/x - B = 0. 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). x_0 is then used to obtain a better approximation; this is done by drawing a tangent line (line 1 in Figure 2-10) to the graph of y = F(x) at the point $[x_0, F(x_0)]$. The intercept of this tangent line becomes the second approximation, x_1. This process is repeated, using tangent line 2 to obtain x_2, and so on.

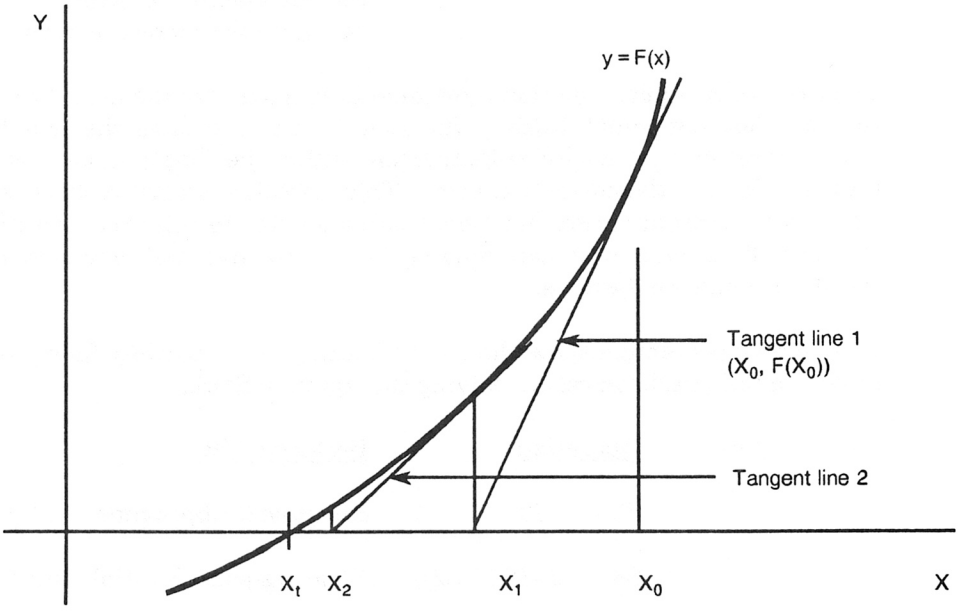


Figure 2-10. Newton's Method of Approximation

The following iteration equation is derived from this process:

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

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

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

Iteration	Operation	Description
1	$x_1 = x_0(2 - x_0B)$	The first approximation is done in the Reciprocal Approximation functional unit and is accurate to 8 bits.
2	$x_2 = x_1(2 - x_1B)$	The second approximation is done in the Reciprocal Approximation functional unit and is accurate to 16 bits.
3	$x_3 = x_2(2 - x_2B)$	The third approximation is done in the Reciprocal Approximation functional unit and is accurate to 30 bits.
4	$x_4 = x_3(2 - x_3B)$	The fourth approximation is done in the Floating-point Multiply functional unit to calculate the correction term.

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

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

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

The reciprocal approximation in Step 1 is correct to 30 bits. By Step 3, it is accurate to 48 bits. This iteration answer is applied as an operand in a full-precision rounded multiply 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	Performed By
1	S3 = 1/S2	Reciprocal Approximation functional unit
2	S6 = S1 * S3	Floating-point Multiply functional unit in half-precision

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

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

The following is another method of computing division:

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

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

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

Double-precision Numbers

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

CPU CONTROL SECTION

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

Exchange Mechanism

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

Exchange Sequence

An exchange sequence occurs before a program can begin running. An exchange sequence performs two simultaneous functions. First, program parameters for the next program are loaded from Central Memory into registers in the CPU. Second, parameters from the previous program are read from the registers and stored back into Central Memory.

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

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

Exchange Package

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

Processor Number Field

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

Memory Error Data Fields

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

- The Syndrome field defines a SECDED error on a memory read or I/O channel.
- The Read Address Bank field defines the bank where a memory read error occurred.
- The Read Error Type field defines the type of memory or I/O error encountered; the error can be either correctable or uncorrectable.
- The Port field defines the port where a memory read or I/O error occurred; these bits are used with the Read Mode bits to identify the operation in error.
- The Read Address Chip Select field identifies the chip on which a memory read error occurred.
- The Read Mode field determines the type of read mode in progress when a memory data error occurred; these bits are used with the Port bits to identify the operation in error.

Program Address Register Field

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

Address Base and Limit Fields

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

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

can be accessed during an instruction fetch sequence. If this absolute address exceeds the limit, a Program Range Error flag is set, which generates an interrupt.

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

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

Register Parity Error Field

The B, T, V, SB, ST, and instruction buffer (IB) registers contain a set of parity bits used for error detection. Each parity bit corresponds to 8 data bits. When a word is written into one of these registers, a set of parity bits is generated and stored with the data bits. When the word is read out of the register, another set of parity bits is generated and compared with the stored set. An error is indicated when the two sets of bits do not match.

Exchange Address Register Field

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

Vector Length Register Field

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

Cluster Number Register Field

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

Flag Register Field

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

The F register contains the following flags:

- Register Parity Error (RPE) flag; set when a parity error occurs during a read operation from a B, T, V, SB, or ST register or from an instruction buffer.
- Interrupt-from-internal CPU (ICP) flag; set when another CPU issues instruction 0014*j*1.
- Deadlock (DL) flag; set when all CPUs in a cluster are holding issue on a Test and Set instruction.
- Programmable Clock Interrupt (PCI) flag; set when the the Programmable Clock reaches a count of 0.
- MCU Interrupt (MCU) flag; set when the Master I/O Processor (MIOP) sends this signal.
- Floating-point Error (FPE) flag; set when a Floating-point Range error occurs in any of the floating-point functional units and the Interrupt-on-floating-point Error (IFP) bit in the M register is set.
- Operand Range Error (ORE) flag; set when a data reference is made outside the boundaries of the DBA and DLA registers and the Interrupt-on-operand Range Error bit is set.
- Program Range Error (PRE) flag; set when an instruction fetch is made outside the boundaries of the IBA and ILA registers.
- Memory Error (ME) flag; set when a correctable or uncorrectable memory error occurs and the corresponding Interrupt-on-memory Error (IME) bit in the M register is set.
- I/O Interrupt (IOI) flag; set when a 6-Mbyte/s channel or a 1,000-Mbyte/s channel completes a transfer.
- Error Exit (EEX) flag; if not in monitor mode or if the Interrupt-in-monitor Mode bit is set, this flag is set by an Error Exit instruction.
- Normal Exit (NEX) flag; if not in monitor mode, this flag is set by a Normal Exit instruction.

Mode Register Field

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

- Enable Second Vector Logical (ESVL) bit; when set, the Second Vector Logical functional unit can be used.
- Program State (PS) bit; this bit is set by the operating system to show whether a CPU, concurrently processing a program with another CPU, is the master or slave in a multitasking situation.
- Floating-point Error Status (FPS) bit; when set, a Floating-point error occurred regardless of the state of the Interrupt-on-floating-point Error bit.
- Bidirectional Memory (BM) bit; when set, block reads and writes can operate concurrently.
- Interrupt-on-operand Range Error (IOR) bit; when set, this bit enables interrupts on Operand Address Range errors.
- Interrupt-on-floating-point Error (IFP) bit; when set, this bit enables interrupts on floating-point errors.
- Interrupt-on-uncorrectable Memory Error (IUM) bit; when set, this bit enables interrupts on uncorrectable memory data errors and on register parity bits.
- Interrupt-on-correctable Memory Error (ICM) bit; when set, this bit enables interrupts on correctable memory data errors.
- Extended Addressing Mode (EAM) bit; when set this bit indicates that 32-bit (Y-mode) addressing takes place. When it is not set, indicates that 24-bit (X-mode) addressing takes place.
- Selected for External Interrupts (SEI) bit; when set, this CPU is preferred for I/O interrupts.
- Interrupt Monitor Mode (IMM) bit; when set, this bit enables all interrupts in monitor mode except PCI, MCU, ICP, and IOI.
- Monitor Mode (MM) bit; when set, this bit inhibits all interrupts except memory errors, normal exit, and error exit.

Vector Not Used Field

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

Waiting for Semaphore Field

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

A Registers Field

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

S Registers Field

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

Instruction Fetch

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

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

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

Instruction Issue

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

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

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

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

Programmable Clock

Each CPU has one Programmable Clock. This 32-bit clock can be loaded with a count value, then decrements one count each CP. An interrupt is generated when the clock reaches a count of 0. These clocks allow the operating system to force interrupts at a particular time or frequency and enhance the use of multitasking in programs.

Performance Monitor

The CRAY Y-MP computer system contains a performance monitor. This monitor consists of several counters which track certain hardware-related events. The following events can be tracked:

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

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

Status Register

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

- Clustered, CLN not set to 0
- Uncorrectable Memory error occurred
- Correctable Memory error occurred
- Program State status
- Floating-point error occurred
- Floating-point interrupt enabled
- Operand range interrupt enabled
- Bidirectional memory enabled
- Processor number count (bits 0 through 2)
- Cluster number count (bits 0 through 3)

SPECIAL FEATURES OF THE CRAY Y-MP COMPUTER SYSTEM

The CRAY Y-MP computer system has several special features that enhance the parallel processing capabilities inherent in all Cray mainframes. Parallel processing can mean different things in different environments; the following subsections discuss parallel processing within a single CPU of a CRAY Y-MP mainframe.

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

Pipelining and Segmentation

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

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

Figure 2-11 shows how a set of elements is pipelined through a segmented Vector functional unit. In the first CP, element 1 of register V1 and element 1 of register V2 enters the first segment of the functional unit. During the next CP, the partial result is moved to the second segment of the functional unit, and element 2 of both Vector registers enters the first segment. This process continues each CP until all elements are completely processed.

In this example, the functional unit is divided into five segments; the functional unit can process up to five different pairs of elements simultaneously. After 5 CPs, the first result leaves the functional unit and enters register V3; subsequent results are available at the rate of one result per CP.

Functional Unit Independence

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

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

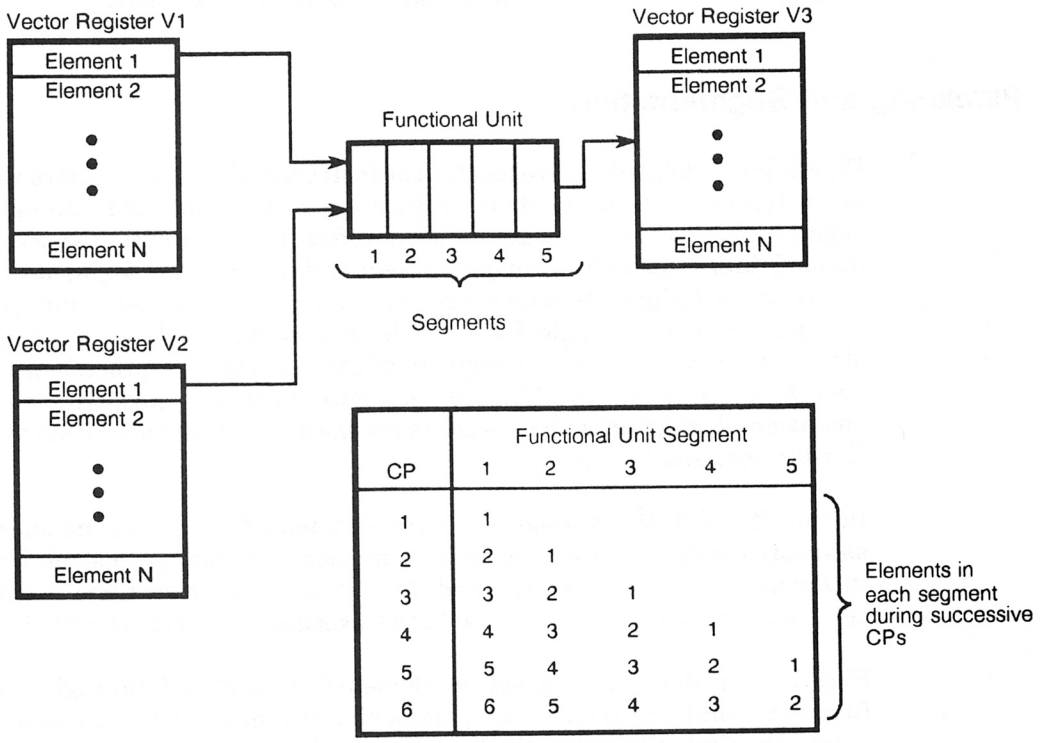


Figure 2-11. Segmentation and Pipelining Example

Vector Processing

One of the most powerful features of the CRAY Y-MP computer system is its vector processing capability. This feature 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. The following subsections describe vector processing, the advantages of using vector processing, and the types of vector instructions.

Definition of Vector Processing

Each CPU of the CRAY Y-MP computer system contains V registers and a number of vector and floating-point functional units that perform vector operations. Refer to

"Vector Registers", "Vector Functional Units", and "Floating-point Functional Units" in this section for more information on these registers and functional units.

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

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

Advantages of Vector Processing

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

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

Vector Chaining

The CRAY Y-MP computer system allows a Vector register reserved for results to become the operand register of a succeeding instruction. This process, called chaining, allows a continuous stream of operands to flow through the vector registers and functional units. Even when a vector load operation pauses due to 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 relationship between the issue time of the chaining instruction and the result data stream. For full chaining to occur, the chaining instruction must have issued and be ready to use element 0 of the result at the same time element 0 arrives at the V register. Partial chaining occurs if the chaining instruction issues after the arrival of element 0.

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

1. Read a vector of integers from memory to Vector register V0.

2. Add the contents of V0 to the contents of V1 and send the results to V2.

3. Shift the results obtained in Step 2 and send the results to V3.

4. Form the logical product of the shifted sum obtained in Step 3 with V4, and send the results to V5.

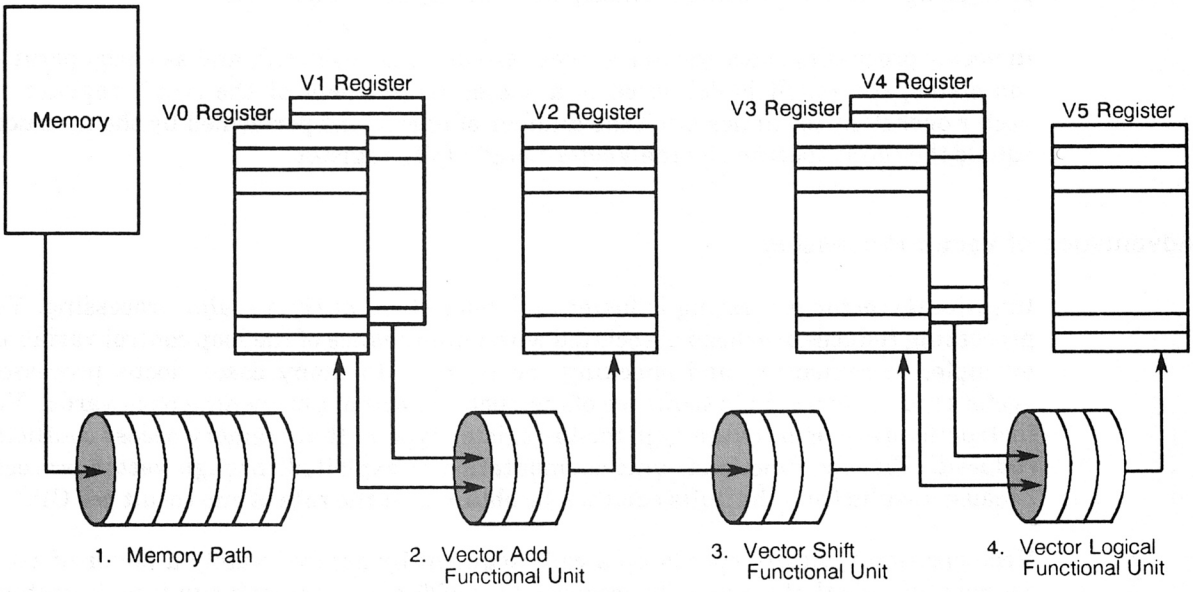


Figure 2-12. Vector Chaining Example

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

Types of Vector Instructions

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

- Vector-vector operand instructions that obtain operand(s) from one or two V registers and enter results into another V register
- Vector-scalar operand instructions that obtain one operand (a constant) from an S register and one operand from a V register and enter results in another V register

- Vector memory instructions that load (read) or store (write) elements to memory
- Vector instructions that set the Vector Mask (VM) register or set/read the Vector Length (VL) register

The vector-vector operand instructions obtain operands from one or two V registers and enter results into another V register. Refer to "Functional Instruction Summary" later in this section for more information on the specific instructions.

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

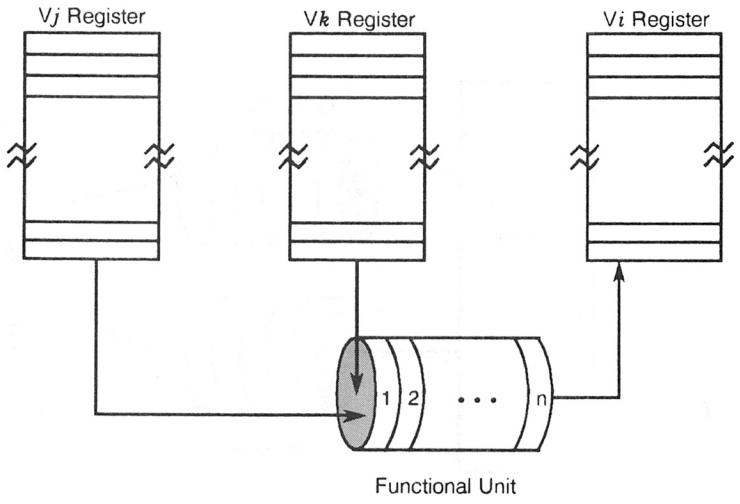


Figure 2-13. Vector-vector Operand Instructions

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

Vector memory instructions transmit data between memory and the V registers (refer to Figure 2-15). A path between memory and the V registers is considered a functional unit for timing considerations. Refer to "Functional Instruction Summary" later in this section for more information on the instructions.

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

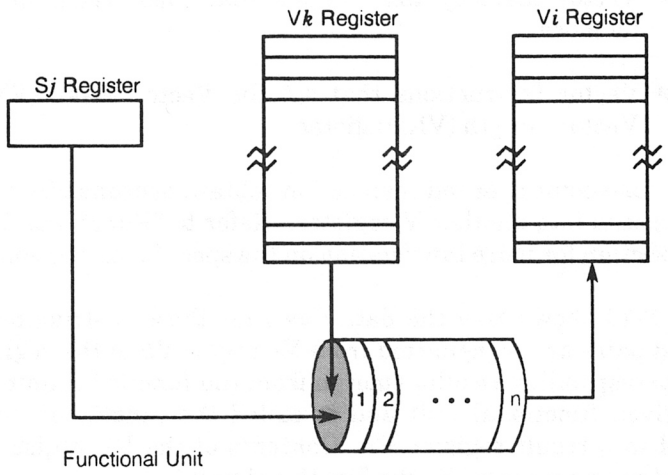


Figure 2-14. Vector-scalar Operand Instructions

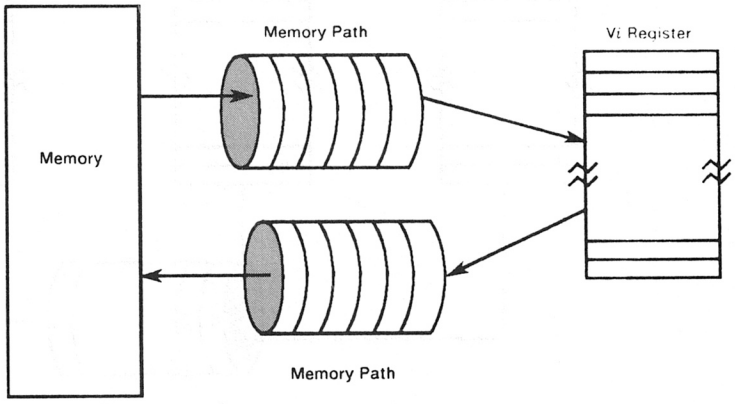


Figure 2-15. Vector Memory Instructions

Most vector memory instructions access memory addresses with a fixed increment value. The Gather and Scatter instructions use two vector registers to gather or scatter elements randomly throughout memory. The first vector register contains the data and the second vector register is used as an index to gather or scatter the data from/to random memory locations.

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

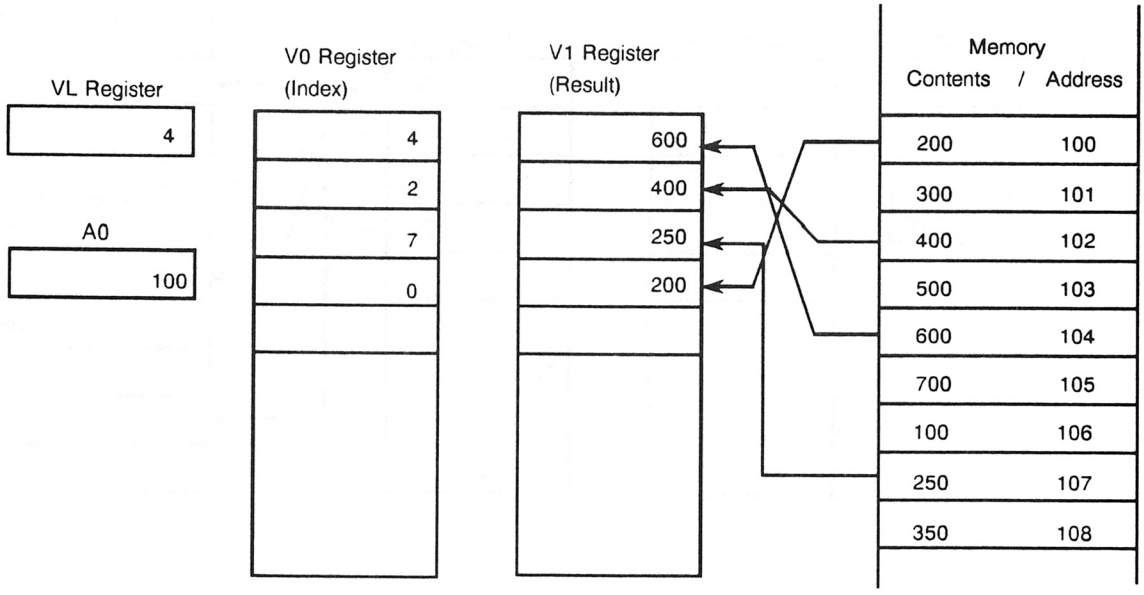


Figure 2-16. Gather Instruction Example

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

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

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

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

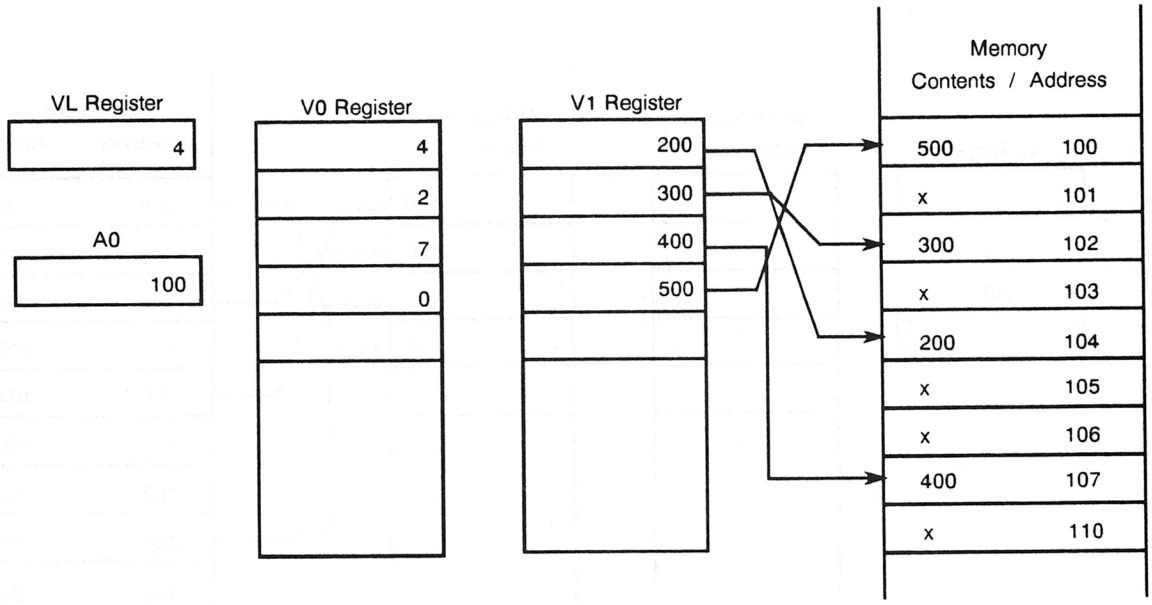


Figure 2-17. Scatter Instruction Example

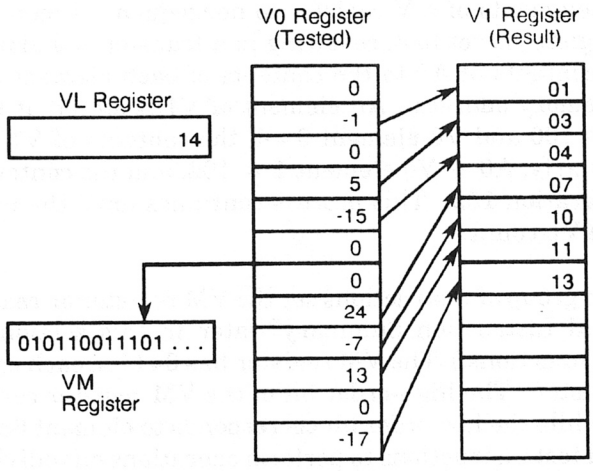


Figure 2-18. Compressed Index Example

CPU INSTRUCTIONS

The following subsections explain the instruction formats, instruction differences between the X-mode and the Y-mode, and special register values used by the CRAY Y-MP computer system. A CPU instruction summary is also included.

Instruction Formats

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

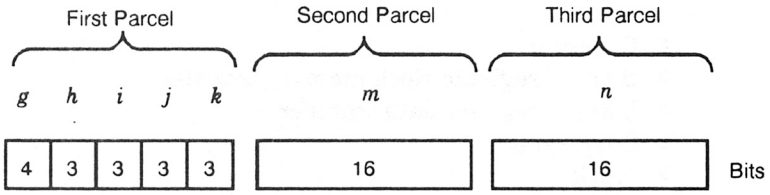


Figure 2-19. General Format for Instructions

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

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

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

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

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

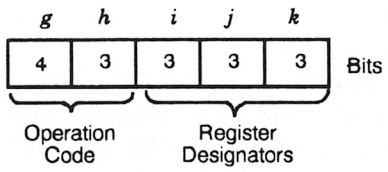


Figure 2-20. 1-parcel Instruction Format with Discrete *j* and *k* Fields

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

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

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

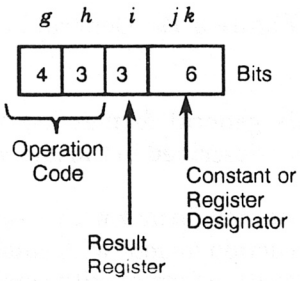


Figure 2-21. 1-parcel Instruction Format with Combined *j* and *k* Fields

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

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

The instruction format used for scalar memory transfers also requires a 22-bit *jkm* field for address displacement. This format uses the 4-bit *g* field for an operation code, the 3-bit *h* field to designate an address index register, and the 3-bit *i* field to designate a source or result register.

Figure 2-22 shows the two general applications for the 2-parcel instruction format with combined *j*, *k*, and *m* fields.

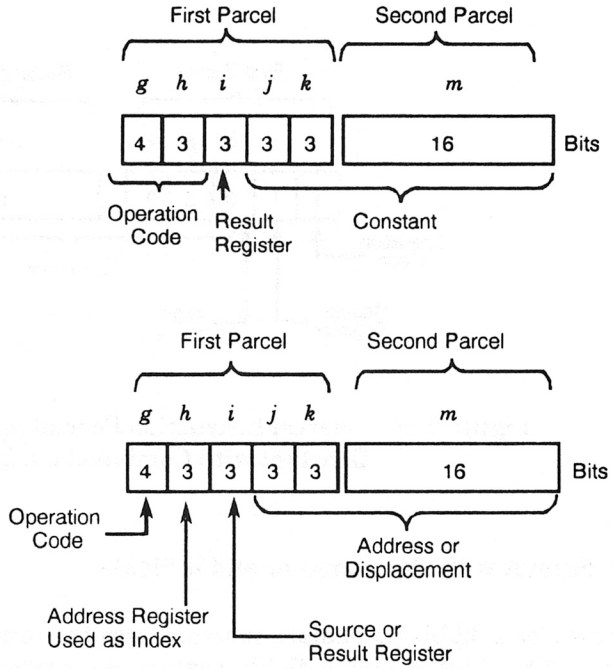


Figure 2-22. 2-parcel Instruction Format with Combined *j*, *k*, and *m* Fields

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

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

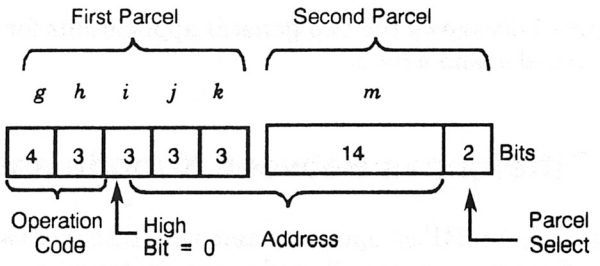


Figure 2-23. 2-parcel Instruction Format with Combined *i*, *j*, *k*, and *m* Fields

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

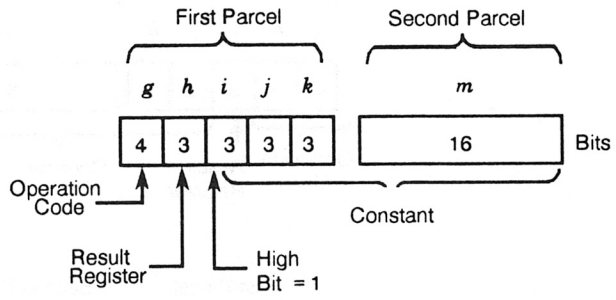


Figure 2-24. 2-parcel Instruction Format for a 24-bit Immediate Constant with Combined *i*, *j*, *k*, and *m* Fields

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

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

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

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

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

Instruction Differences Between X-mode and Y-mode

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

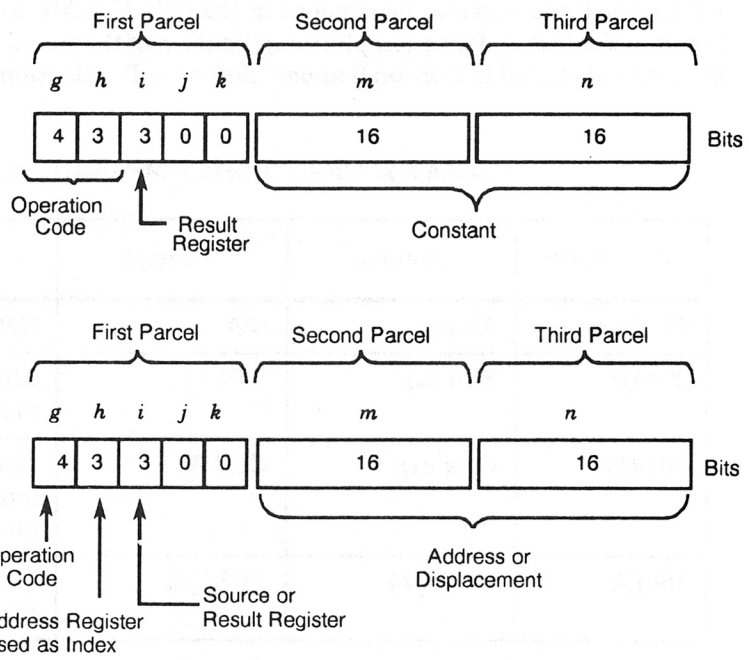


Figure 2-25. 3-parcel Instruction Format with Combined *m* and *n* Fields

Table 2-1. CRAY Y-MP 3-parcel Instruction

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

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

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

Table 2-2. CRAY Y-MP/X-MP Instruction Differences

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

Special Register Values

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

Table 2-3. Special Register Values

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

Monitor Mode Instructions

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

Special CAL Syntax Forms

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

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

```
VL A0
VL 1
```

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

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

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

CPU Instruction Summary

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

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

In some instructions, register designators are prefixed by the following letters that have special meaning to the assembler. The letters and their meaning are listed as follows.

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

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

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

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

Functional Units Instruction Summary

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

Functional Unit	Instructions
Address Add (Integer)	030, 031
Address Multiply (Integer)	032
Scalar Add (Integer)	060, 061
Scalar Logical	042-051
Scalar Shift	052-055, 056, 057
Scalar Pop/Parity/	026
Leading Zero	027
Vector Add (Integer)	154-157
Vector Logical	140-147, 175
Second Vector Logical	140-145
Vector Shift	150, 151, 153, 152
Vector Pop/Parity	174*ij*1, 174*ij*2
Floating-point Add	062, 063, 170-173
Floating-point Multiply	064-067, 160-167
Floating-point Reciprocal	070, 174*ij*0

Functional Unit	Instructions
Memory (Scalar)	100-130
Memory (Vector)	176, 177

Functional Instruction Summary

This subsection summarizes the instruction by the function 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. For more information on these instructions, refer to the manuals listed in Section 6 under "Software Publications".

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

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

Register Entry Instructions

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

Transfers Into A Registers

The following instructions transmit values into the A registers.

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

Transfers Into S Registers

The following instructions transmit values into the S registers.

Machine Instruction	CAL Syntax	Description
040*ijkm*[4,5] or 041*ijkm*[4,5]	S*i* *exp*	Transmit *exp* into S*i* (040) or Transmit one's complement of *exp* into S*i* (041)
040*i*00*mn*[4,5] or 041*i*00*mn*[4,5]	S*i* *exp*	Transmit *exp* into S*i* (040) or Transmit one's complement of *exp* into S*i* (041)
042*i*00[2]	S*i* -1	Enter -1 into S*i*
042*ijk*	S*i* <*exp*	Form one's mask in S*i* *exp* bits from right; *jk* field gets 64-*exp*
042*ijk*[2]	S*i* #>*exp*	Form zeros mask in S*i* *exp* bits from left; *jk* field gets *exp*
042*i*77[2]	S*i* 1	Enter 1 into S*i*
043*i*00[2]	S*i* 0	Clear S*i*
043*ijk*	S*i* >*exp*	Form one's mask in S*i* *exp* bits from left; *jk* field gets *exp*
043*ijk*[2]	S*i* #<*exp*	Form zeros mask in S*i* *exp* bits from right; *jk* field gets 64-*exp*
047*i*00[2]	S*i* #SB	Enter one's complement of sign bit into S*i*
051*i*00[2]	S*i* SB	Enter sign bit into S*i*
071*i*30	S*i* 0.6	Transmit (0.75×2^{48}) as normalized floating-point constant into S*i*
071*i*40	S*i* 0.4	Transmit 0.4 as normalized floating-point constant into S*i*
071*i*50	S*i* 1.	Transmit 1.0 as normalized floating-point constant into S*i*
071*i*60	S*i* 2.	Transmit 2.0 as normalized floating-point constant into S*i*
071*i*70	S*i* 4.	Transmit 4.0 as normalized floating-point constant into S*i*

Transfers Into V Registers

The following instructions transmit values into the V registers.

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

Transfers Into Semaphore Register

The following instructions transmit values into the Semaphore registers.

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

Inter-register Transfer Instructions

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

Transfers to A Registers

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

Machine Instruction	CAL Syntax	Description
023*ij*0	A*i* S*j*	Transmit (S*j*) to A*i*
023*i*01	A*i* VL	Transmit (VL) to A*i*
024*ijk*	A*i* B*jk*	Transmit (B*jk*) to A*i*
026*ij*7	A*i* SB*j*	Transmit (SB*j*) to A*i*
030*i*0k^2	A*i* A*k*	Transmit (A*k*) to A*i*
031*i*0k^2	A*i* -A*k*	Transmit negative of (A*k*) to A*i*
033*i*00	A*i* CI	Channel number of highest priority interrupt request to A*i* (*j* = 0)

Machine Instruction	CAL Syntax	Description
033*ij*0	A*i* CA,A*j*	Current address of channel (A*j*) to A*i* ($j \neq 0$, $k=0$)
033*ij*1	A*i* CE,A*j*	Error flag of channel (A*j*) to A*i* ($j \neq 0$, $k=1$)

Transfers to S Registers

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

Machine Instruction	CAL Syntax	Description
025*ijk*	B*jk* A*i*	Transmit (A*i*) to B*jk*
027*ij*7	SB*j* A*i*	Transmit (A*i*) to SB*j*
047*i*0*k*[2]	S*i* #S*k*	Transmit one's complement of (S*k*) to S*i*
051*i*0*k*[2]	S*i* S*k*	Transmit (S*k*) to S*i*
061*i*0*k*[2]	S*i* -S*k*	Transmit negative of (S*k*) to S*i*
071*i*0*k*	S*i* A*k*	Transmit (A*k*) to S*i* with no sign extension
071*i*1*k*	S*i* +A*k*	Transmit (A*k*) to S*i* with sign extension
071*i*2*k*	S*i* +FA*k*	Transmit (A*k*) to S*i* as unnormalized floating-point number
072*i*00	S*i* RT	Transmit (RTC) to S*i*
072*i*02	S*i* SM	Transmit semaphore to S*i*
072*ij*3	S*i* ST*j*	Transmit (ST*j*) register to S*i*
073*i*00	S*i* VM	Transmit (VM) to S*i*
073*i*01	S*i* SR*j*	Transmit (SR*j*) to S*i* ($j=0$)
073*ij*3	ST*j* S*i*	Transmit (S*i*) to ST*j*
074*ijk*	S*i* T*jk*	Transmit (T*jk*) to S*i*
075*ijk*	T*jk* S*i*	Transmit (S*i*) to T*jk*
076*ijk*	S*i* V*j*,A*k*	Transmit (V*j* element (A*k*)) to S*i*

Transfers to V Registers

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

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

Transfer to Vector Mask Register

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

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

Transfer to Vector Length Register

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

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

Transfer to Semaphore Register

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

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

Memory Transfer Instructions

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

Bidirectional Memory Transfers

The following instructions enable or disable bidirectional memory transfers.

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

Memory References

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

Machine Instruction	CAL Syntax	Description
002700	CMR	Complete memory references

Stores

The following instructions store values into memory.

Machine Instruction	CAL Syntax	Description
035*ijk*	,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
035*ijk*[2]	0,A0 B*jk*,A*i*	Store (A*i*) words from B registers starting at register *jk* to memory starting at address (A0)
037*ijk*	,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
037*ijk*[2]	0,A0 T*jk*,A*i*	Store (A*i*) words from T registers starting at register *jk* to memory starting at address (A0)
11*hijkm*[5]	*exp*,A*h* A*i*	Store (A*i*) to ((A*h*) + *exp*)
11*hi00mn*[6]	*exp*,A*h* A*i*	Store (A*i*) to ((A*h*) + *exp*)
11*hi*000[2,5]	,A*h* A*i*	Store (A*i*) to (A*h*)

Machine Instruction	CAL Syntax	Description
11*hi*0000[2,6]	,A*h* A*i*	Store (A*i*) to (A*h*)
110*ijkm*[2,5]	*exp*,0 A*i*	Store (A*i*) to *exp*
110*i*00*mn*[2,6]	*exp*,0 A*i*	Store (A*i*) to *exp*
110*ijkm*[2,5]	*exp*, A*i*	Store (A*i*) to *exp*
110*i*00*mn*[2,6]	*exp*, A*i*	Store (A*i*) to *exp*
13*hijkm*[5]	*exp*,A*h* S*i*	Store (S*i*) to ((A*h*) + *exp*)
13*hi*00*mn*[6]	*exp*,A*h* S*i*	Store (S*i*) to ((A*h*) + *exp*)
130*ijkm*[2,5]	*exp*,0 S*i*	Store (S*i*) to *exp*
130*i*00*mn*[2,6]	*exp*,0 S*i*	Store (S*i*) to *exp*
130*ijkm*[2,5]	*exp*, S*i*	Store (S*i*) to *exp*
130*i*00*mn*[2,6]	*exp*, S*i*	Store (S*i*) to *exp*
13*hi*000[2,5]	,A*h* S*i*	Store (S*i*) to (A*h*)
13*hi*0000[2,6]	,A*h* S*i*	Store (S*i*) to (A*h*)
1770*jk*	,A0,A*k* V*j*	Store (V*j*) to memory starting at (A0) increased by (A*k*)
1770*j*0	,A0,1 V*j*	Store (V*j*) to memory in consecutive addresses starting with (A0)
1771*jk*	,A0,V*k* V*j*	Store (V*j*) to memory using memory address (A0) + (V*k*)

Loads

The following instructions load values from memory.

Machine Instruction	CAL Syntax	Description
034*ijk*	B*jk*,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at address *jk*
034*ijk*[2]	B*jk*,A*i* 0,A0	Load (A*i*) words from memory starting at address (A0) to B registers starting at address *jk*
036*ijk*	T*jk*,A*i* ,A0	Load (A*i*) words from memory starting at address (A0) to T registers starting at address *jk*

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

Integer Arithmetic Instructions

Integer arithmetic operations obtain operands from registers and return results to registers. No direct memory references are 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, whether 24-bit, 32-bit, or 64-bit, is two's complement and is represented as such in the registers. The Address Add and Address Multiply functional units perform 24-bit (X-mode) and 32-bit (Y-mode) arithmetic. The Scalar Add functional unit and the Vector Add functional unit perform 64-bit arithmetic. No overflow is detected by functional units when performing integer arithmetic.

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

24-bit or 32-bit Integer Arithmetic

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

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

64-bit Integer Arithmetic

The following instructions perform 64-bit integer arithmetic.

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

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

Floating-point Arithmetic Instructions

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

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

Floating-point Range Errors

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

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

Floating-point Addition and Subtraction

The following instructions perform floating-point addition or subtraction.

Machine Instruction	CAL Syntax	Description
062*ijk*	S*i* S*j*+FS*k*	Floating-point sum of (S*j*) and (S*k*) to S*i*
062*i*0*k*[2]	S*i* +FS*k*	Normalize (S*k*) to S*i*
063*ijk*	S*i* S*j*-FS*k*	Floating-point difference of (S*j*) and (S*k*) to S*i*
063*i*0*k*[2]	S*i* -FS*k*	Transmit the normalized negative of (S*k*) to S*i*

Machine Instruction	CAL Syntax	Description
170*ijk*	V*i* S*j*+FV*k*	Floating-point sums of (S*j*) and (V*k* elements) to V*i* elements
170*i*0*k*^{2}	V*i* + FV*k*	Transmit normalized (V*k* elements) to V*i* elements
171*ijk*	V*i* V*j*+FV*k*	Floating-point sums of (V*j* elements) and (V*k* elements) to V*i* elements
172*ijk*	V*i* S*j*-FV*k*	Floating-point differences of (S*j*) and (V*k* elements) to V*i* elements
172*i*0*k*^{2}	V*i* -FV*k*	Transmit normalized negative of (V*k* elements) to V*i* elements
173*ijk*	V*i* V*j*-FV*k*	Floating-point differences of (V*j* elements) and (V*k* elements) to V*i* elements

Floating-point Multiplication

The following instructions perform floating-point multiplication.

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

Reciprocal Iteration

The following instructions perform reciprocal iteration operations.

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

Reciprocal Approximation

The following instructions perform floating-point reciprocal approximation operations.

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

Logical Operation Instructions

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

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

Logical Products

The following instructions produce logical products.

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

Logical Sums

The following instructions produce logical sums.

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

Logical Differences

The following instructions produce logical differences.

Machine Instruction	CAL Syntax	Description
046*ijk*	S*i* S*j*\S*k*	Logical difference of (S*j*) and (S*k*) to S*i*

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

Logical Equivalence

The following instructions produce logical equivalence.

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

Vector Mask

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

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

Machine Instruction	CAL Syntax	Description
175*ij*7	V*i*,VM V*j*,M	Set VM bits and register V*i* to V*j*, for negative elements of V*j*

Merge

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

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

Shift Instructions

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

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

Machine Instruction	CAL Syntax	Description
057*ijk*	S*i* S*j*,S*i*>A*k*	Shift (S*j*) and (S*i*) right by (A*k*) places to (S*i*)
057*ij*0[2]	S*i* S*j*,S*i*>1	Shift (S*j*) and (S*i*) right one place to (S*i*)
057*i*0*k*[2]	S*i* S*i*>A*k*	Shift (S*i*) right (A*k*) places to S*i*
150*ijk*	V*i* V*j*<A*k*	Shift (V*j* elements) left by (A*k*) places to V*i* elements
150*ij*0[2]	V*i* V*j*<1	Shift (V*j* elements) left one place to V*i* elements
151*ijk*	V*i* V*j*>A*k*	Shift (V*j* elements) right by (A*k*) places to V*i* elements
151*ij*0[2]	V*i* V*j*>1	Shift (V*j* elements) right one place to V*i* elements
152*ijk*	V*i* V*j*,V*j*<A*k*	Double shift of (V*j* elements) left (A*k*) places to V*i* elements
152*ij*0[2]	V*i* V*j*,V*j*<1	Double shift of (V*j* elements) left one place to V*i* elements
153*ijk*	V*i* V*j*,V*j*>A*k*	Double shift of (V*j* elements) right (A*k*) places to V*i* elements
153*ij*0[2]	V*i* V*j*,V*j*>1	Double shift of (V*j* elements) right one place to V*i* elements

Bit Count Instructions

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

Scalar Population Count

The following instruction performs the scalar population count.

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

Vector Population Count

The following instruction performs the vector population count.

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

Population Count Parity

The following instructions perform population parity count.

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

Scalar Leading Zero Count

The following instruction performs leading zero count.

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

Branch Instructions

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

Unconditional Branch Instructions

The following instructions perform unconditional branch operations.

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

Conditional Branch Instructions

The following instructions perform conditional branch operations.

Machine Instruction	CAL Syntax	Description
010*ijkm*	JAZ *exp*	Jump to *exp* if (A0) = 0 (*i*2 = 0)
011*ijkm*	JAN *exp*	Jump to *exp* if (A0) ≠ 0 (*i*2 = 0)
012*ijkm*	JAP *exp*	Jump to *exp* if (A0) positive; includes (A0) = 0 (*i*2 = 0)
013*ijkm*	JAM *exp*	Jump to *exp* if (A0) negative (*i*2 = 0)
014*ijkm*	JSZ *exp*	Jump to *exp* if (S0) = 0 (*i*2 = 0)
015*ijkm*	JSN *exp*	Jump to *exp* if (S0) ≠ 0 (*i*2 = 0)
016*ijkm*	JSP *exp*	Jump to *exp* if (S0) positive; includes (S0) = 0 (*i*2 = 0)
017*ijkm*	JSM *exp*	Jump to *exp* if (S0) negative (*i*2 = 0)

Return Jump

The following instruction performs a return jump operation.

Machine Instruction	CAL Syntax	Description
007*ijkm*	R *exp*	Return jump to *exp*; set B00 to (P) + 2

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*[1]	CA,A*j* A*k*	Set the CA register for the channel indicated by (A*j*) to (A*k*) and activate the channel
001000	PASS	Pass
0011*jk*[1]	CL,A*j* A*k*	Set the CL register for the channel indicated by (A*j*) to (A*k*) address
0012*j*0[1]	CI,A*j*	Clear the interrupt flag and error flag for the channel indicated by (A*j*); clear device master-clear (output channel)
0012*j*1[1]	MC,A*j*	Clear the interrupt flag and error flag for the channel indicated by (A*j*); set device master-clear (output channel); clear device ready-held (input channel)
0013*j*0[1]	XA A*j*	Enter XA register with (A*j*)

Set Real-time Clock

The following instruction performs a Real-time Clock operation.

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

Programmable Clock Interrupt Instructions

The following instructions perform Programmable Clock operations.

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

Interprocessor Interrupt Instructions

The following instructions perform Interprocessor Interrupt operations.

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

Cluster Number Instructions

The following instruction sets the cluster number.

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

Operand Range Error Interrupt Instructions

The following instructions enable or disable Operand Range Error interrupts.

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

Performance Counters

The following instructions select maintenance features of the performance monitor.

Machine Instruction	CAL Syntax	Description
0015*j*0[1,3]		Select performance monitor
001501[1,3]		Disable Port A error correction
001511[1,3]		Disable Port B error correction
001521[1,3]		Disable Port C error correction
001531[1,3]		Enable T register data to be routed through Port D error correction instead of Port B
001541[1,3]		Enables replacement of checkbyte with data on ports C and D for writes and replacement of data with checkbytes on ports A, B, and D for reads
001541[1,3]		Enable replacement of checkbyte with V*k* data on Port C during execution of instruction 1771*jk*
073*i*11[1,3]		Read performance counter into S*i*
073021[1,3]		Increment performance counter
073031[1,3]		Clear all maintenance modes
073061[1,3]		Increment current performance counter (lower)

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

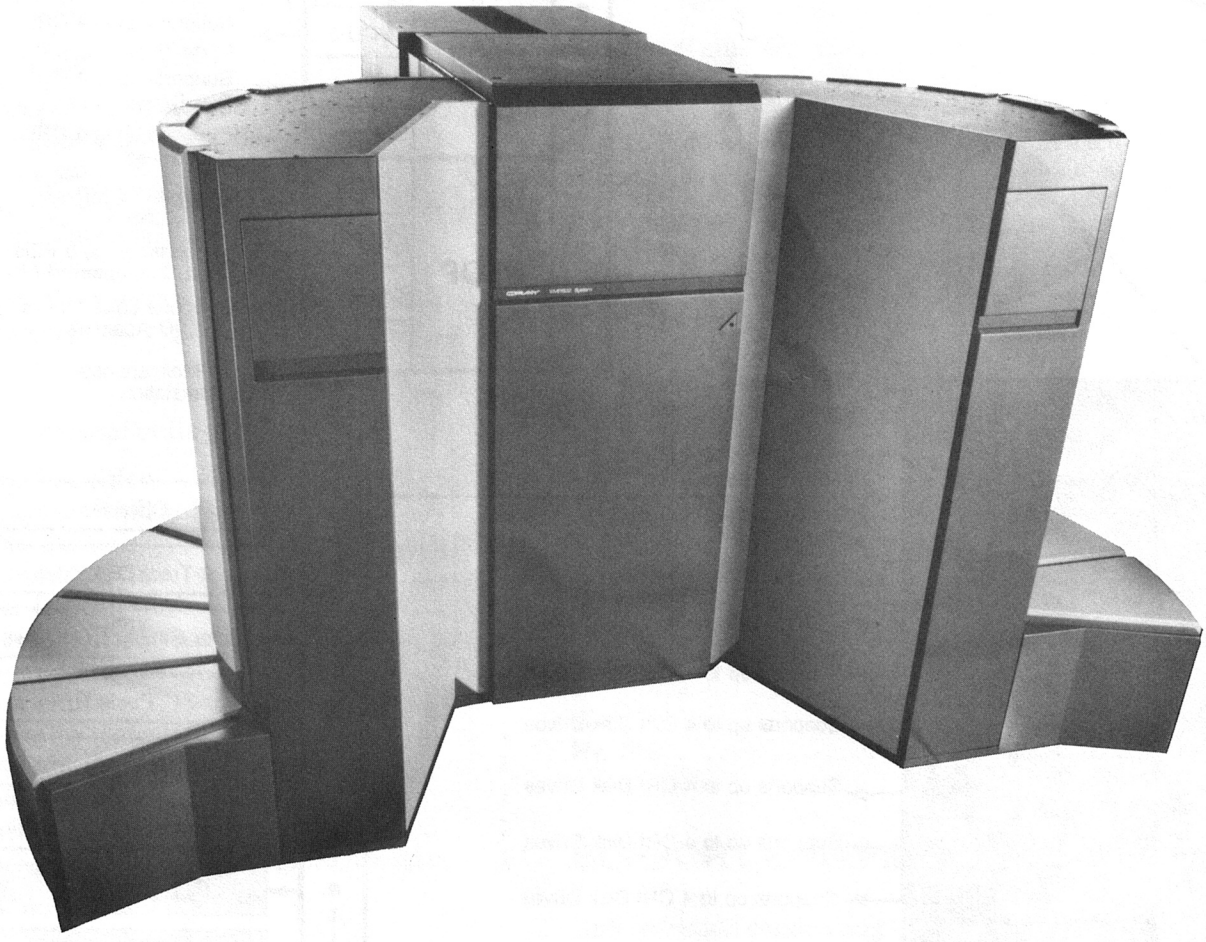
Footnote	Description
1	Privileged to monitor mode
2	Special syntax mode
3	Not supported by CAL Version 2
4	Generated depending on the value of exp
5	X-mode only
6	Y-mode only

CRAY Y-MP8 COMPUTER SYSTEM

Model Numbers

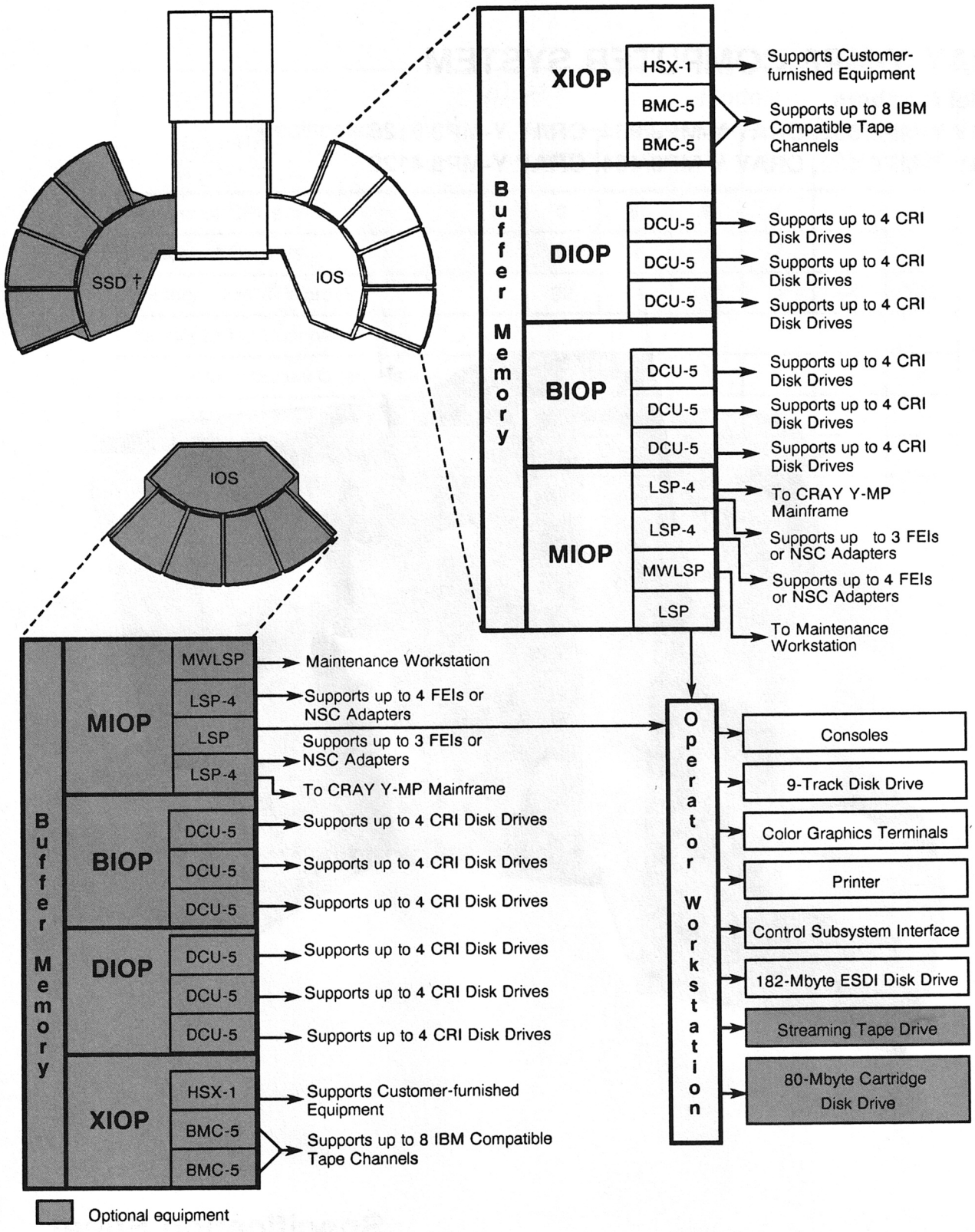
CRAY Y-MP8/832, CRAY Y-MP8/864, CRAY Y-MP8/8128

CRAY Y-MP8/432, CRAY Y-MP8/464, CRAY Y-MP8/4128



Specification Sheet

CRAY RESEARCH, INC.



† Optional equipment. (The CRAY Y-MP8 computer system may be configured with the following model numbers: SSD-3I, SSD-5I, SSD-5, SSD-6, and SSD-7. If the system is configured with an SSD-3I or SSD-5I, an IOS/Integrated SSD chassis will replace the SSD chassis shown above.)

CRAY Y-MP8 Computer System Maximum Configuration

CRAY Y-MP8 SPECIFICATIONS

CRAY Y-MP8 MAINFRAME FEATURES

System Clock

Speed ..... 6.0 ns

CPU Specifications

Number of CPUs 4, 8

Number of registers per CPU:

- Address (A) registers, 32 bits each 8
- Intermediate address (B) registers, 32 bits each 64
- Scalar (S) registers, 64 bits each 8
- Intermediate scalar (T) registers, 64 bits each 64
- Vector (V) registers, 64 bits per element, 64 elements per register 8

Number of functional units per CPU:

- Address addition 1
- Address multiplication 1
- Scalar addition 1
- Scalar shift 1
- Scalar logical 1
- Scalar population/parity leading zero 1
- Vector addition 1
- Vector shift 1
- Full vector logical 1
- 2nd vector logical 1
- Floating-point addition 1
- Floating-point multiplication 1
- Floating-point reciprocal approximation . 1

Shared Resources

I/O section:

- 1000-Mbyte/s channels 2
- 100-Mbyte/s channels 4, 8
- 6-Mbyte/s channels 4, 8

Shared Resources (continued)

Central memory:

- Word width ..... 64 bits
- SECDED error correction ..... 8 bits
- Memory size ..... 32, 64, 128 Mwords
- Number of banks 256
- Number of modules 32
- Number of ports per CPU 4

Number of clusters 7, 9

Number of shared registers contained in each cluster:

- Shared address (SB) registers, 32 bits each (Y-mode), 24 bits each (X-mode) 8
- Shared scalar (ST) registers, 64 bits each 8
- Semaphore (SM) registers, 1 bit each 32

Real-time clock (64 bits) 1

PHYSICAL DESCRIPTION

Floor space ..... 16 ft^2 (1.5 m^2)
Weight ..... 5,400 lbs (2,450 kg)
Height ..... 6.4 ft (1.91 m)

SUPPORT EQUIPMENT

Refrigeration condensing unit 1
Motor-generator sets 1-3
Operator workstation 1
Maintenance workstation 1
Heat exchanger unit 1

For individual CRAY Y-MP8 model specifications, refer to the table on the following page.

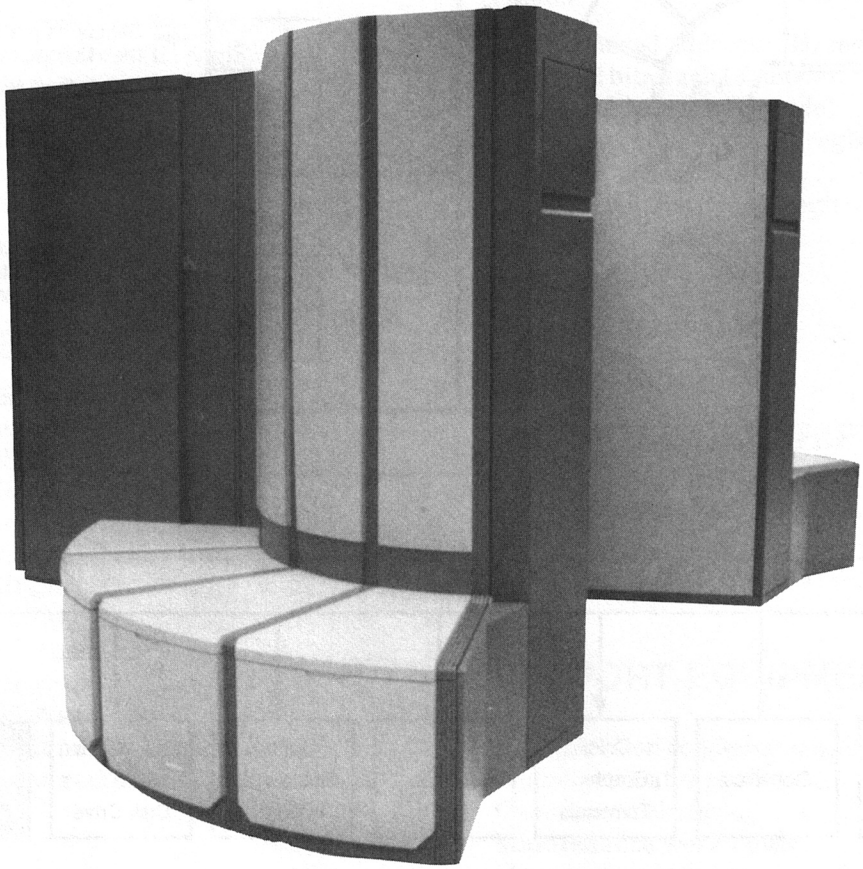
CRAY Y-MP8 MODEL SPECIFICATIONS

Specifications	Models					
	832	864	8128	432	464	4128
Number of CPUs	8	8	8	4	4	4
Number of Clusters	9	9	9	7	7	7
Memory Size (MWords)	32	64	128	32	64	128
Number of I/O Channels:						
1000-MByte/s Channels	2	2	2	2	2	2
100-MByte/s Channels	8	8	8	4	4	4
6-MByte/s Channels	8	8	8	4	4	4

CRAY Y-MP4 COMPUTER SYSTEM

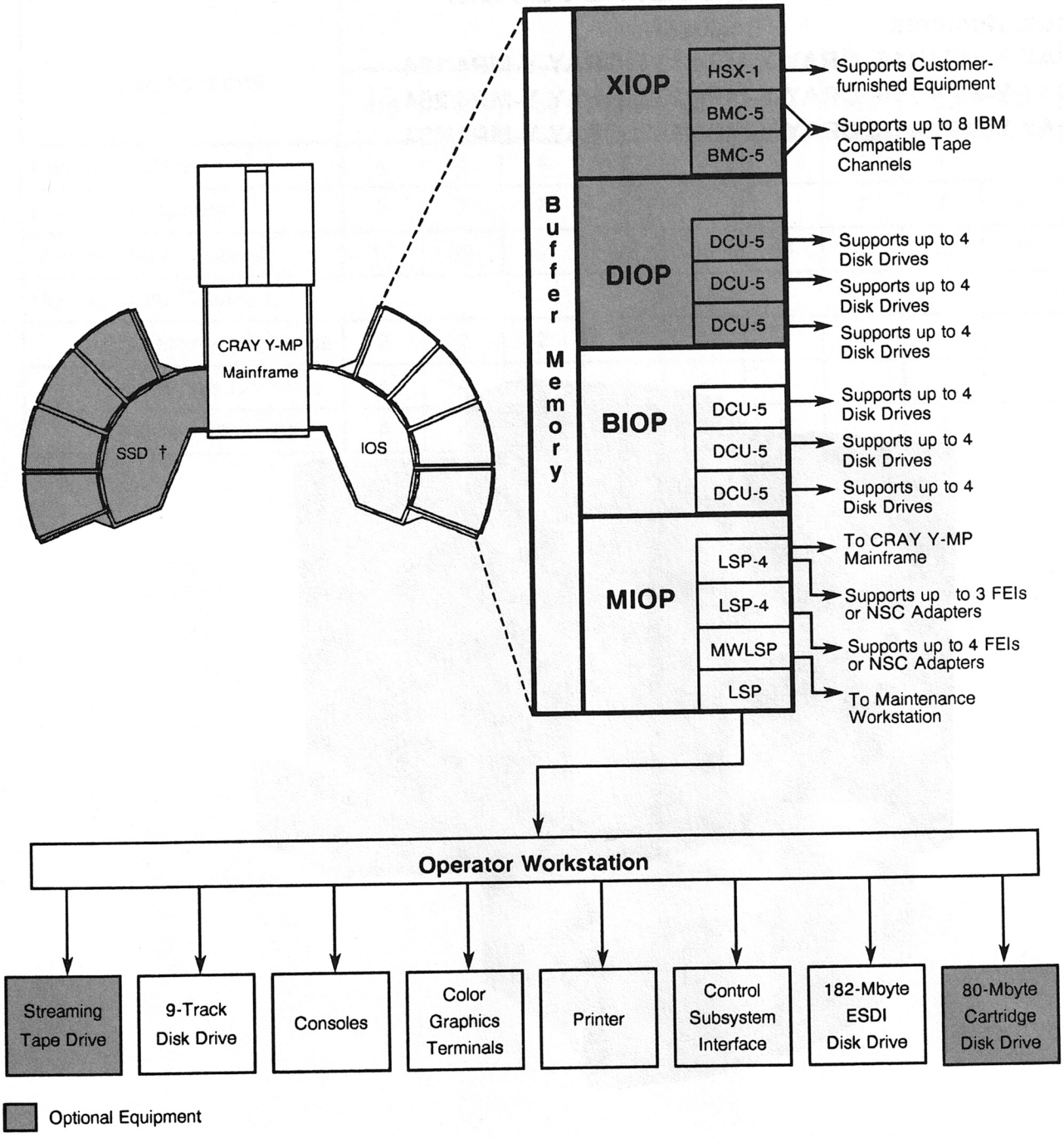
Model Numbers

CRAY Y-MP4/116, CRAY Y-MP4/132, CRAY Y-MP4/164
CRAY Y-MP4/216, CRAY Y-MP4/232, CRAY Y-MP4/264
CRAY Y-MP4/416, CRAY Y-MP4/432, CRAY Y-MP4/464



CRAY RESEARCH, INC.

Specification Sheet



† Optional equipment. (The CRAY Y-MP8 computer system may be configured with the following model numbers: SSD-3I, SSD-5I, SSD-5, SSD-6, and SSD-7. If the system is configured with an SSD-3I or SSD-5I, an IOS/Integrated SSD chassis will replace the SSD footprint shown above.)

CRAY Y-MP4 Computer System Maximum Configuration

CRAY Y-MP4 SPECIFICATIONS

CRAY Y-MP4 MAINFRAME FEATURES

System Clock

Speed ..... 6.0 ns

CPU Specifications

Number of CPUs 1, 2, 4

Number of registers per CPU:

- Address (A) registers, 32 bits each 8
- Intermediate address (B) registers, 32 bits each 64
- Scalar (S) registers, 64 bits each 8
- Intermediate scalar (T) registers, 64 bits each 64
- Vector (V) registers, 64 bits per element, 64 elements per register 8

Number of functional units per CPU:

- Address addition 1
- Address multiplication 1
- Scalar addition 1
- Scalar shift 1
- Scalar logical 1
- Scalar population/parity leading zero 1
- Vector addition 1
- Vector shift 1
- Full vector logical 1
- 2nd vector logical 1
- Floating-point addition 1
- Floating-point multiplication 1
- Floating-point reciprocal approximation . 1

Shared Resources

Central memory:

- Word width ..... 64 bits
- SECDED error correction ..... 8 bits
- Memory size ..... 16, 32, 64 Mwords
- Number of banks 128
- Number of modules 16
- Number of ports per CPU 4

Shared Resources (continued)

I/O section:

- 1000-Mbyte/s channels 0, 1, 2
- 100-Mbyte/s channels 1, 2, 4
- 6-Mbyte/s channels 1, 2, 4

Number of clusters 7

Number of shared registers contained in each cluster:

- Shared address (SB) registers, 32 bits each (Y-mode), 24 bits each (X-mode) 8
- Shared scalar (ST) registers, 64 bits each 8
- Semaphore (SM) registers, 1 bit each 32

Real-time clock (64 bits) 1

PHYSICAL DESCRIPTION

Floor space ..... 16 ft^2 (1.5 m^2)
Weight ..... 5,400 lbs (2,450 kg)
Height ..... 6.4 ft (1.9 m)

SUPPORT EQUIPMENT

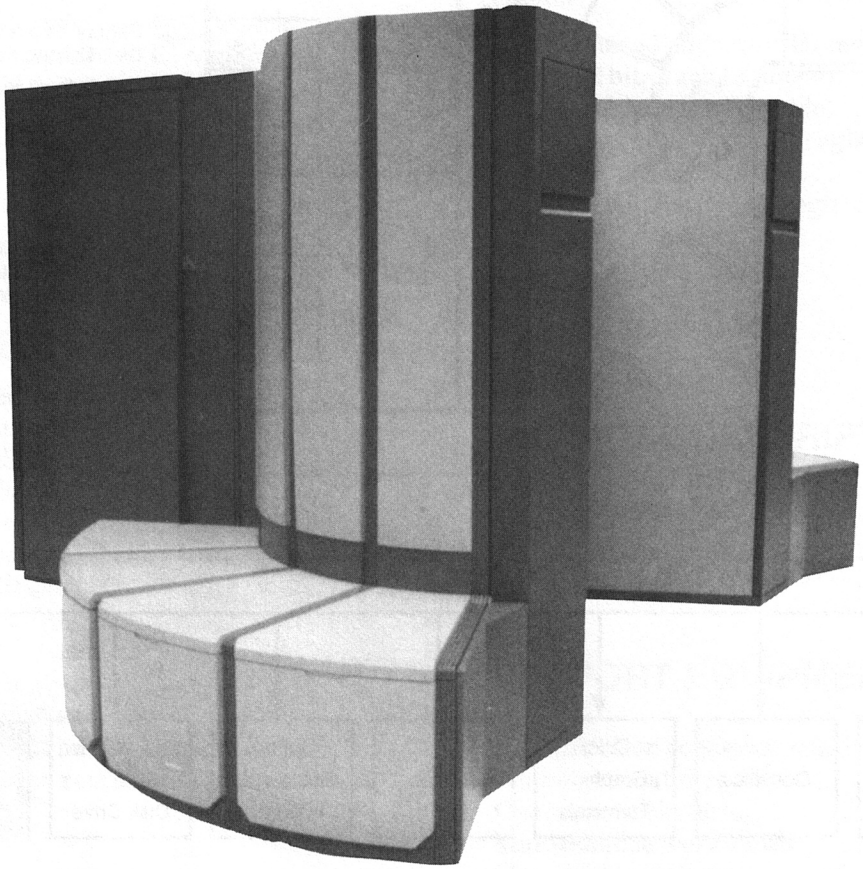
Refrigeration condensing unit 1
Motor-generator sets 1-2
Operator workstation 1
Maintenance workstation 1
Heat exchanger unit 1

For individual CRAY Y-MP4 model descriptions, refer to the table on the following page.

CRAY Y-MP4 COMPUTER SYSTEM

Model Numbers

CRAY Y-MP4/116, CRAY Y-MP4/132, CRAY Y-MP4/164
CRAY Y-MP4/216, CRAY Y-MP4/232, CRAY Y-MP4/264
CRAY Y-MP4/416, CRAY Y-MP4/432, CRAY Y-MP4/464

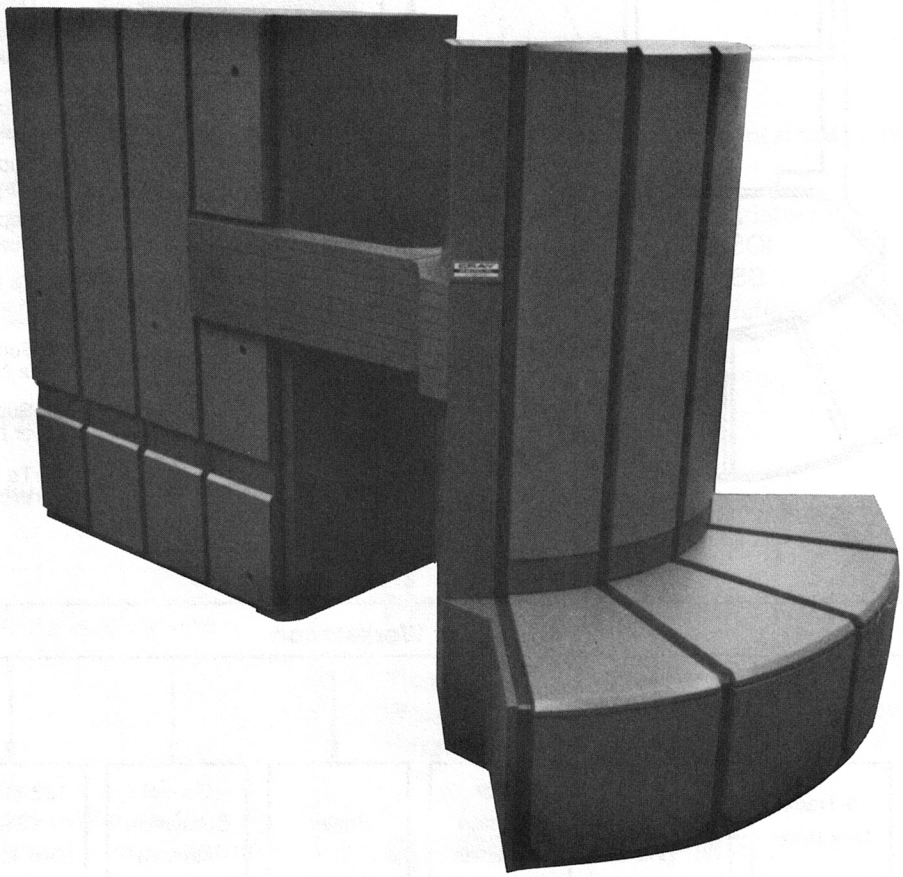


CRAY RESEARCH, INC.

Specification Sheet

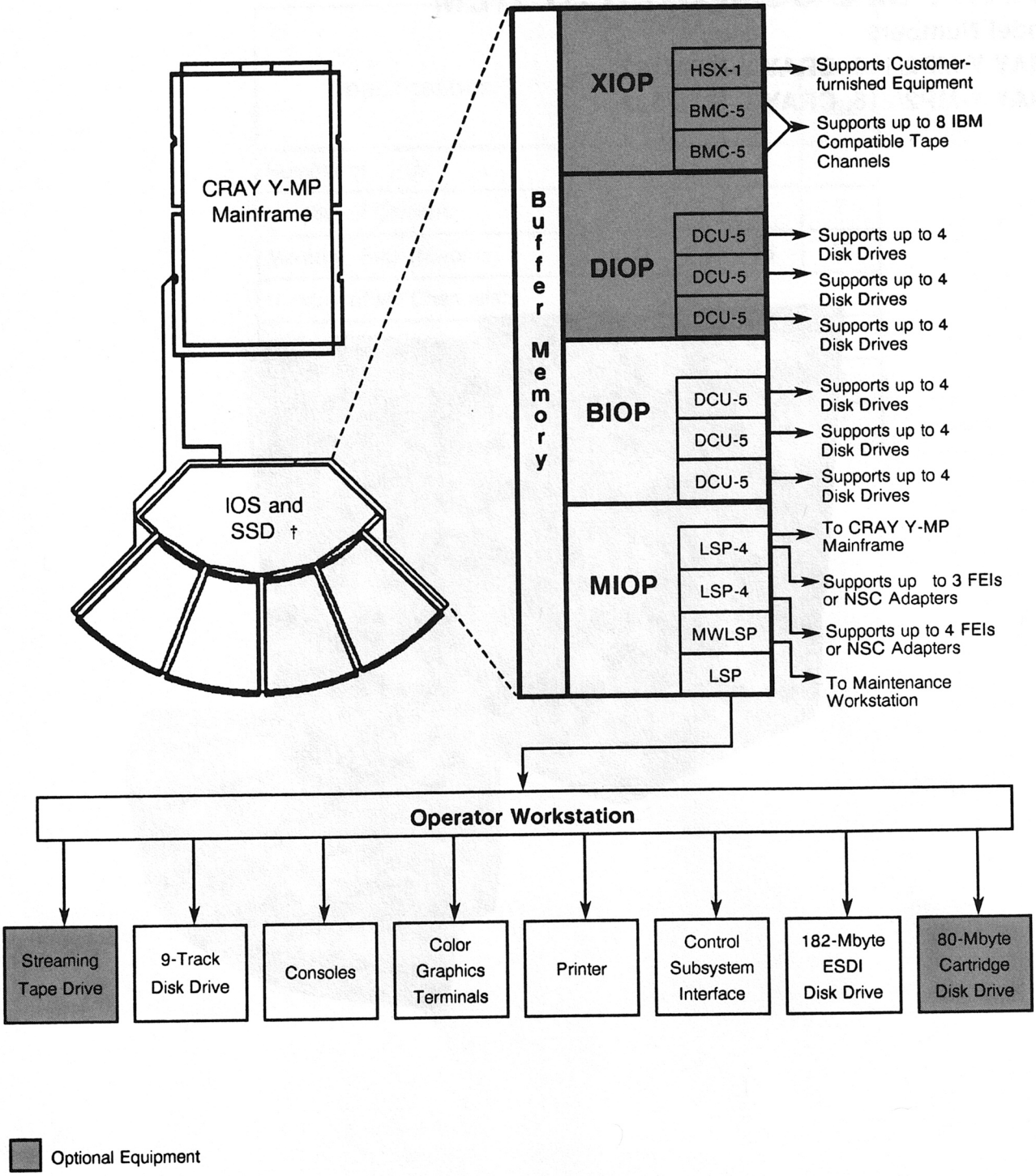
CRAY Y-MP2 COMPUTER SYSTEM

Model Numbers
CRAY Y-MP2/116, CRAY Y-MP1/132
CRAY Y-MP2/216, CRAY Y-MP2/232



CRAY RESEARCH, INC.

Specification Sheet



† Optional equipment. (The CRAY Y-MP2 computer system can be configured with the following model numbers: SSD-3I and SSD-5I.)

CRAY Y-MP2 Computer System Maximum Configuration

CRAY Y-MP2 SPECIFICATIONS

CRAY Y-MP2 MAINFRAME FEATURES

System Clock

Speed ..... 6.0 ns

CPU Specifications

Number of CPUs 1, 2

Number of registers per CPU:

- Address (A) registers, 32 bits each 8
- Intermediate address (B) registers, 32 bits each 64
- Scalar (S) registers, 64 bits each 8
- Intermediate scalar (T) registers, 64 bits each 64
- Vector (V) registers, 64 bits per element, 64 elements per register 8

Number of functional units per CPU:

- Address addition 1
- Address multiplication 1
- Scalar addition 1
- Scalar shift 1
- Scalar logical 1
- Scalar population/parity leading zero 1
- Vector addition 1
- Vector shift 1
- Full vector logical 1
- 2nd vector logical 1
- Floating-point addition 1
- Floating-point multiplication 1
- Floating-point reciprocal approximation . 1

Shared Resources

I/O section:

- 1000-Mbyte/s channel 0, 1
- 100-Mbyte/s channels 1, 2
- 6-Mbyte/s channels 1, 2

Shared Resources (continued)

Central memory:

- Word width ..... 64 bits
- SECDED error correction ..... 8 bits
- Memory size ..... 16, 32 Mwords
- Number of banks 64
- Number of modules 8
- Number of ports per CPU 4

Number of clusters 7

Number of shared registers contained in each cluster:

- Shared address (SB) registers, 32 bits each (Y-mode), 24 bits each (X-mode) 8
- Shared scalar (ST) registers, 64 bits each 8
- Semaphore (SM) registers, 1 bit each 32

Real-time clock (64 bits) 1

PHYSICAL DESCRIPTION

Floor space ..... 13.7 ft^2 (1.3 m^2)
Weight ..... 2,500 lbs (1,100 kg)
Height ..... 6.4 ft (1.9 m)

SUPPORT EQUIPMENT

Refrigeration condensing unit 1
Motor-generator sets 1
Operator workstation 1
Maintenance workstation 1
Heat exchanger unit 1

For individual CRAY Y-MP2 model specifications, refer to the table on the following page.

CRAY Y-MP2 MODEL SPECIFICATIONS

Specifications	Models			
	216	232	116	132
Number of CPUs	2	2	1	1
Number of Clusters	7	7	7	7
Memory Size (Mwords)	16	32	16	32
Number of I/O Channels:				
1000-Mbyte/s Channels	1	1	0	0
100-Mbyte/s Channels	2	2	1	1
6-Mbyte/s Channels	2	2	1	1

CONTENTS

3 - I/O SUBSYSTEM 3-1

I/O Processors 3-1

I/O Processor Functions 3-3

I/O Channel Interfaces 3-3

I/O Subsystem Buffer Memory 3-4

System Operator Workstation 3-4

I/O Subsystem Model D Specification Sheet 3-5

3 - I/O SUBSYSTEM

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

The IOS for the CRAY Y-MP2 computer system is housed in its own stand-alone cabinet. Figure 1-2 shows the IOS chassis (IOC). For the CRAY Y-MP8 or CRAY Y-MP4 computer systems, the standard IOS forms one leg of the Y-shaped configuration. A second IOS, optional with the CRAY Y-MP8 computer system, must be located within 9.5 ft (2.89 m.) of the mainframe.

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

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

I/O PROCESSORS

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

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

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

Instructions are stored in Local Memory and are transferred into the Instruction stack under the control of the Program Address counter. Instructions issue from the Instruction stack and are then decoded into the appropriate control signals.

The Program Exit stack of the control section stores return addresses for program subroutine calls. The registers provide nested levels of subroutines in a program. A register keeps track of the levels involved.

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

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

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

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

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

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

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

The memory section unique to each IOP, called Local Memory, consists of random access solid-state storage. Refer to the specification sheet at the end of this section for the different memory sizes available. An error correction and detection network ensures that the data written into memory or read from memory is correct.

I/O Processor Functions

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

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

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

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

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

Another IOP that can be added to interface with additional Disk Storage Units (DSUs) is called the Disk I/O Processor (DIOP). The DIOP connects to Buffer Memory and to the mainframe's Central Memory over a 100-Mbyte/s channel pair. The DIOP data transfer sequence is similar to the BIOP's sequence.

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

I/O Channel Interfaces

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

I/O SUBSYSTEM BUFFER MEMORY

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

SYSTEM OPERATOR WORKSTATION

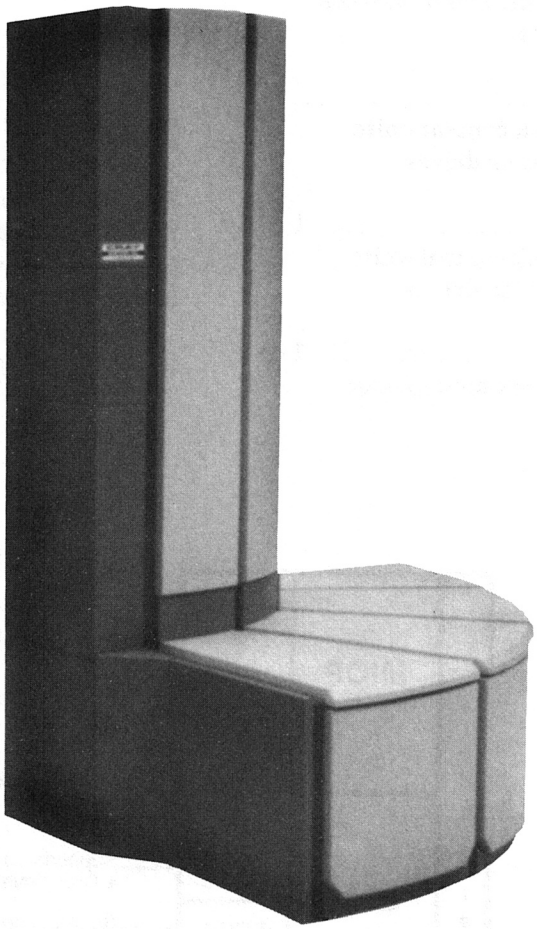
VMEbus technology is used on the System Operator Workstation (OWS) and replaces the Peripheral Expander and system consoles used on CRAY X-MP computer systems. The OWS is a microcomputer system that provides the following functions:

- System operator interface
- System deadstart and master clear functions
- Software maintenance utilities
- Local tape, and local printer
- System time-of-day clock

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

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

I/O SUBSYSTEM MODEL D



CRAY RESEARCH, INC.

Specification Sheet

IOS MODEL D SPECIFICATIONS

IOS MODEL D FEATURES

System Clock

Speed ..... 12.5 ns

IOP Specifications

Maximum number of IOPs 4

Number of MIOPs 1
The MIOP supports the front-end interfaces (FEIs) and station software.

Number of BIOPs 1
The BIOP supports the disk control units (DCU-5) that support the disk drives.

Number of DIOPs (optional) ..... 1 or 2
The DIOP supports the disk control units (DCU-5) that support the disk drives.

Number of XIOPs (optional) ..... 1 or 2
The XIOP supports the block multiplexer channel (BMC-5).

IOP Specifications (continued)

Memory:

- Word width ..... 64 bits
- Buffer memory size † ..... 4 Mwords
- Local memory size per CPU ... 64 Kparcels

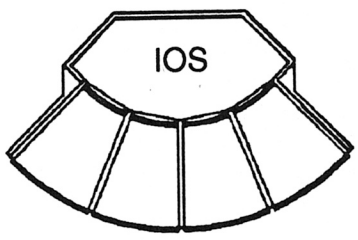
PHYSICAL DESCRIPTION

Floor Space ..... 15 ft² (1.4 m²)
Weight ..... 3,290 lbs, (1,492 Kgs)

SUPPORT EQUIPMENT

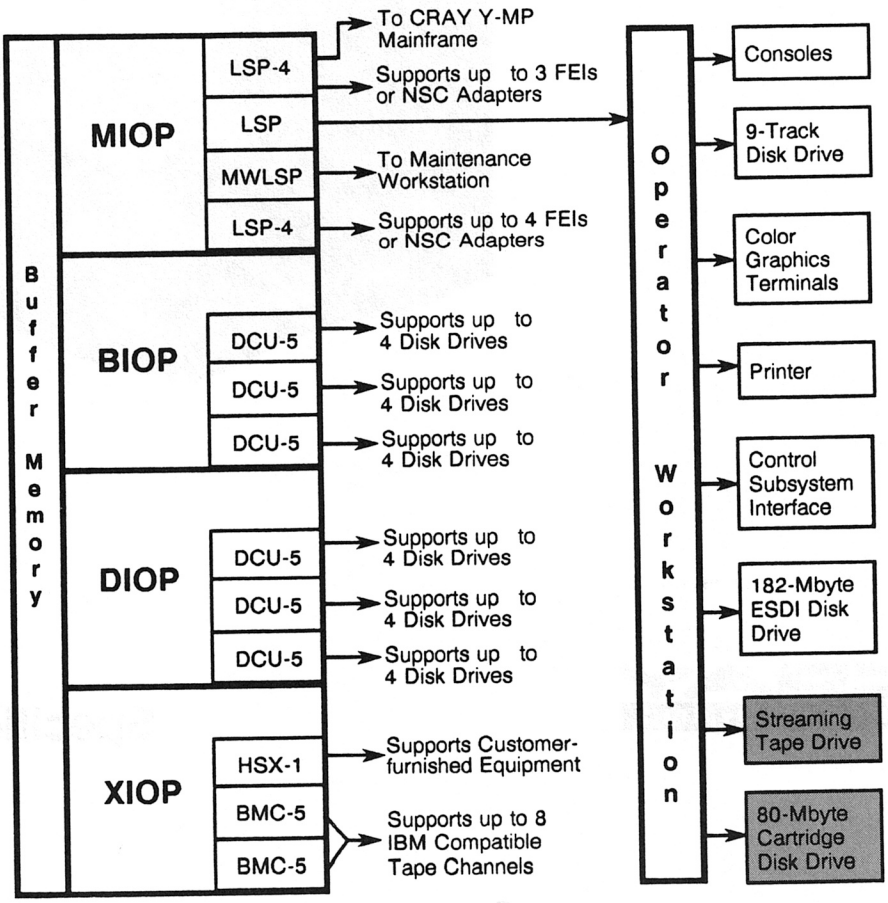
Power distribution unit 1
Refrigeration condensing unit 1
Maintenance workstation 1
Motor-generator set 1

† Buffer memory size can be upgraded to 8 or 32 MWords.



IOS Model D Maximum Configuration

Optional Equipment



CONTENTS

4 - SSD SOLID-STATE STORAGE DEVICE 4-1
SSD Functions 4-1
SSD Memory Size 4-2
SSD Memory Transfer and Data Protection 4-2
SSD Solid-state Storage Device Specification Sheet 4-3

4 - SSD SOLID-STATE STORAGE DEVICE

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

The SSD chassis is shown in Figure 1-3. This self-contained chassis holds the SSD memory modules, channels, and control logic. The power supplies and cooling system are similar to those used in a Cray mainframe. The SSD-3I and SSD-5I are special versions of the SSD that are housed within the IOS chassis (IOC).

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

SSD FUNCTIONS

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

Data is transferred between the SSD buffers and the SSD memory, and between the SSD buffers and the CRAY Y-MP mainframe. For data transfer, each Central Processing Unit (CPU) memory port handles 16 words through one of two buffers. Each CPU memory port handles every other word to or from memory

The SSD has five memory ports, which are defined in the following list.

- Port 0 controls refreshing of SSD memory
- Port 1 connects the SSD to a maintenance computer through a 6-Mbyte/s channel with asynchronous control logic
- Port 2 provides up to four 100-Mbyte/s channels that may be connected to the IOS
- Ports 3 and 4 connect the SSD directly to the CRAY Y-MP mainframe using a channel that has a maximum data transfer rate of 1,000-Mbyte/s.

SSD MEMORY SIZE

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

SSD MEMORY TRANSFER AND DATA PROTECTION

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

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

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

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

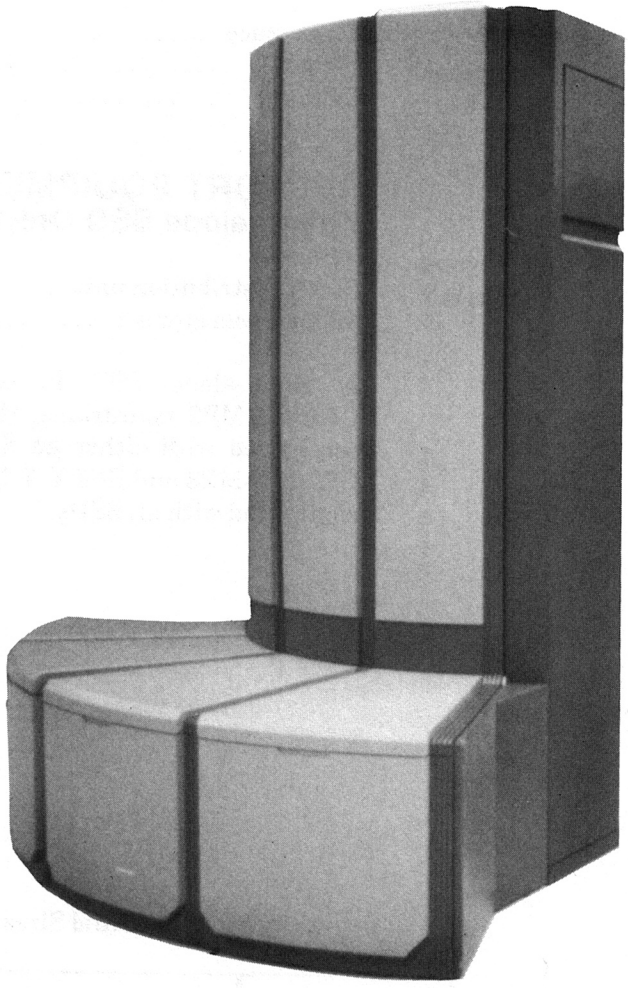
If an error occurred, the 8 bits of the checkbyte are analyzed by the SECDED logic to find the number of altered bits. If only a single bit was altered, the correction logic resets that bit to the correct state and passes the corrected word out to either the mainframe or the IOS. The Error Logger receives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

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

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

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

SSD SOLID-STATE STORAGE DEVICE



CRAY RESEARCH, INC.

Specification Sheet

SSD SPECIFICATIONS

SSD FEATURES

Storage word size ..... 72 bits
(64 data bits and 8 check bits)

Minimum block size ..... 64 words

Maximum blocks per operation:

- Port 2 ..... 16 Kwords
- Ports 3 and 4 ..... 16 Mwords

Data transfer burst speeds:

- Port 2 ..... 100 Mbyte/s
- Ports 3 and 4 ..... 1,000 Mbyte/s

Available channels:

- Port 1 (6 Mbyte/s) 1
- Port 2 (100 Mbyte/s) 4
- Port 3 (1,000 Mbyte/s) 1
- Port 4 (1,000 Mbyte/s) † 1

Maximum band width ..... 26.9 Gbits/s

Port priorities: lowest port number has highest priority.

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

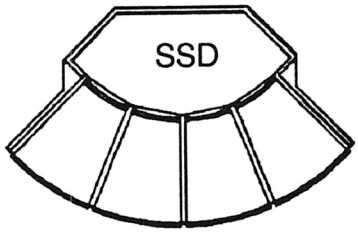
PHYSICAL DESCRIPTION (Stand-alone SSD Only)

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

SUPPORT EQUIPMENT (Stand-alone SSD Only)

Power distribution unit 1
Motor-generator set 1

A stand-alone SSD is not offered with the CRAY Y-MP2 mainframe; this model may only be configured with either an SSD-3I or SSD-5I. The CRAY Y-MP8 and CRAY Y-MP4 mainframes can be configured with all SSDs.



SSD Models and Sizes

SSD Model	Size (Mwords)	Size (Mbytes)
SSD-3I††	32	512
SSD-5I††	128	1,024
SSD-5	128	1,024
SSD-6	256	2,048
SSD-7	512	4,096

† The SSD-3I and SSD-5I do not support port 4.

†† The SSD-3I and SSD-5I are housed in the IOS D chassis.

CONTENTS

5 - PERIPHERAL EQUIPMENT 5-1
DISK CONTROLLER UNITS AND DISK STORAGE UNITS 5-1
Disk Controller Units 5-1
Disk Storage Units 5-1
DS-40 Disk Subsystem 5-2
DS-40 Disk Subsystem Standard Configurations 5-3
DS-40D Disk Daisy Chain Configurations 5-3
DS-41 Disk Subsystem 5-3
DS-41 Disk Subsystem Standard Configurations 5-4
DS-41A Disk Subsystem Field-upgradable Configurations 5-4
DS-41D Disk Daisy Chain Configurations 5-4
DS-41R Disk Subsystem Redundant Configurations 5-5
DD-49 Disk Storage Unit 5-5
NETWORK INTERFACES 5-6
FEI-1 Front-end Interface 5-6
Fiber-optic Link 5-6
FEI-3 Front-end Interface 5-7
Direct Network Connections 5-7
High-speed External (HSX) Communications Channel 5-7
High Performance Parallel Interface (HiPPI) 5-7
DEC VAX Supercomputer Gateway 5-8
DS-40 AND DS-40D DISK SUBSYSTEMS SPECIFICATION SHEET 5-9
DS-41, DS-41D, AND DS-41R DISK SUBSYSTEMS SPECIFICATION SHEET 5-11
DD-49 DISK DRIVE SPECIFICATION SHEET 5-13
FRONT-END INTERFACE SPECIFICATION SHEET 5-15
FOL-3 FIBER-OPTIC LINK SPECIFICATION SHEET 5-17

5 - PERIPHERAL EQUIPMENT

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

DISK CONTROLLER UNITS AND DISK STORAGE UNITS

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

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

Disk Controller Units

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

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

Disk Storage Units

The DSUs store data on magnetic disks. Sector slipping mechanisms are provided so that the operating system has fewer flaws to keep track of.

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

The recording surface available to each head group is called a disk track, which is the basic storage unit reserved by the operating system. Each disk track has sectors where

data is recorded and read back. The data in one sector is called a data block and includes verification and error-correction data. Data can be transferred between an IOP's Local Memory and the disk surface only in these data blocks.

The following subsections describe the specific DSUs.

DS-40 Disk Subsystem

The DS-40 Disk Subsystem consists of the following components: the DD-40 Disk Storage Unit (DSU), the DC-40 Disk Control Unit (DCU), and the Disk Controller Cabinet (DCC-2). The DD-40 contains four spindles 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 the DS-40 Disk Subsystem Specification Sheet at the end of this section for the exact configuration information. Each physical disk drive (spindle) consists of six rotating platters and ten recording surfaces. Data is accessed by 19 read/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 parcels (512 64-bit words) of IOP data plus verification and error-correction data. Data can be transferred between the IOP's Local Memory and the disk surface only in blocks of this fixed size. Sectors may be chained for both read and write operations.

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

- Controls up to 8 spindles (two DD-40 Disk Drives)
- Passes control functions to the selected drives
- Passes status from the drives to the DCU-5
- Buffers read and write data for transfers between DCU-5 and the disk drives
- Generates error-correction codes for write data
- Checks read data correction codes and corrects read data when necessary
- Controls distribution of read/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/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/write operations. A defect parameter becomes part of the sector ID field when the drive is formatting.

DS-40 Disk Subsystem Standard Configurations

The DS-40 Disk Subsystem standard configuration has the following components:

- One Disk Controller Cabinet (DCC-2)
- Four DC-40 Disk Controllers (housed in the DCC-2)
- Four DD-40 Disk Storage Units (DSUs)

DS-40D Disk Daisy Chain Configurations

The DS-40D disk daisy chain configuration has the following components:

- One Disk Controller Cabinet (DCC-2)
- Four DC-40 Disk Controllers (housed in the DCC-2)
- Eight DD-40 Disk Storage Units (DSUs)

The DS-40D disk daisy chain configuration doubles the capacity (not the performance) of the DS-40 Disk Subsystem without adding another channel or controller. An additional DD-40 Disk Storage Unit is connected to the second port of the DC-40, which is a dual-ported interface. Only one port can be active at any given time.

DS-41 Disk Subsystem

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 nine rotating platters and fifteen recording surfaces. Data is accessed by 15 read/write heads. A servo mechanism, which controls the read/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/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-correction data. Data can be transferred between the IOP's Local Memory and the disk surface 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 DCU-5 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 DCU-5
- Buffers read and write data for transfers between DCU-5 and the disk drives
- Generates error-correction codes for write data
- Checks read data correction codes and corrects read data when necessary
- Controls distribution of read/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/write head reaches the field of 0's, the head skips over the flawed area of the sector over written 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/write operations. A defect parameter becomes part of the sector ID field when the drive is formatting.

DS-41 Disk Subsystem Standard Configurations

A standard DS-41 Disk Subsystem configuration has the following components:

- One Disk Storage Cabinet (DE-41)
- Four DD-41 Disk Drives (housed in the DE-41)
- One Disk Controller Cabinet (DCC-2A)
- Four DC-40 Disk Controllers (housed in the DCC-2A)

DS-41A Disk Subsystem Field-upgradable Configurations

A field-upgradable DS-41A Disk Subsystem configuration has the following components:

- One Disk Storage 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. Up to three DS-41Bs can be installed in a DS-41A Disk Subsystem.

DS-41D Disk Daisy Chain Configurations

A daisy chain DS-41D configuration has the following components:

- Two Disk Storage Cabinets (DE-41s)
- Eight DD-41 disk drives (housed in the two DE-41s)
- One Disk Controller Cabinet (DCC-2A)
- Four DC-41 Disk Controllers (housed in the DCC-2A)

A DS-41D disk daisy chain configuration doubles the capacity (not the performance) of the DS-41 without adding another channel or controller. Each DC-41 contains a dual-ported interface with only one port active at a time in a daisy chain configuration. A second DE-41 with four DD-41 disk drives is daisy chained with the other DD-41s in a DS-41D configuration.

DS-41R Disk Subsystem Redundant Configurations

A redundant DS-41R configuration has the following components:

- Two Disk Storage Cabinets (DE-41s)
- Eight DD-41 disk drives (housed in the two DE-41s)
- Two Disk Controller Cabinets (DCC-2As)
- Eight DC-41 Disk Controllers (housed in the two DCC-2As)

A DS-41R redundant configuration uses two DCC-2A Disk Controller Cabinets to provide the system with dual channel access. All DD-41 Disk Drives can be accessed from either of the two DCC-2A controller cabinets. If the primary channel path is not in service, the additional channel path provides access to all DD-41s in the system.

DD-49 Disk Storage Unit

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

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

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

The DD-49 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 recreated. If a second sector in a track becomes flawed, the process is repeated. If a third sector in a track becomes flawed, the operating system must mark the sector as unavailable. Sector slipping takes place off-line. A hardware diagnostic reformats the track with slipped sectors.

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

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

NETWORK INTERFACES

The CRAY Y-MP computer system can be connected to a wide variety of computer systems (often referred to as "front-end systems") and networks, either directly from the mainframe or through the IOS. This enables users of non-Cray computer systems to exploit the CRAY Y-MP system's extraordinary computational power. The following subsections describe the methods used to interface the CRAY Y-MP computer system with other computer systems and networks.

FEI-1 Front-end Interface

The FEI-1 front-end interface provides communication between the Cray mainframe and many different types of front-end computer systems. The FEI-1 compensates for differences in channel widths, machine word size, electrical logic levels, and control protocols. Refer to the Front-end Interface Specification Sheet at the end of this section for a complete list of compatible mainframes and minicomputers.

The FEI-1 is housed in a stand-alone cabinet located near the host computer. The cabinet is air-cooled and operates directly from the AC power mains; power consumption varies with each type of interface. Internal power supplies provides 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 programmer.

Fiber-optic Link

The Cray Research, Inc. (CRI) fiber-optic link (FOL) is used as a channel extender for 6-Mbyte/s (16-bit asynchronous) channels. It replaces the conventional wire cabling between a Cray computer system and an FEI-1 with fiber-optic cables. Fiber-optic cabling enhances the performance of the FEI-1 by eliminating the occasional problems related to system isolation, including induced noise, variable ground potentials, and radio frequency radiation found in wire cabling. Fiber-optic cabling overcomes these problems and, in addition, provides a secure link for transmitting data over distances up to 3280 ft (1000 m).

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

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

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

FEI-3 Front-end Interface

The FEI-3 is a family of front-end interfaces that enables certain VME-based microcomputers and workstations to communicate with a Cray system over a standard 6 Mbyte/s I/O channel. Specific FEI-3 applications depend on the capabilities of the VME workstations or microcomputers. For example, CRI uses the FEI-3 to connect Cray systems to an Operator Workstation based on the Motorola Delta microcomputer. Some other possible FEI-3 applications are listed below.

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

Each FEI-3 interface consists of two VME-compatible circuit boards that install into the target VME system, plus supporting cables and software drivers. The customer furnishes and provides support for the target VME system.

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

Direct Network Connections

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

High-speed External (HSX) Communications Channel

The Cray HSX is a special high-speed external communications channel that provides full duplex, point-to-point communications between a CRAY Y-MP computer system and a very fast, user-supplied device. The channel operates at up to 100 Mbytes/s and could be used, for instance, to network multiple Cray systems or to communicate with a very fast graphics output device. The HSX hardware is located in the Auxiliary I/O Processor (XIOP) of the IOS.

High Performance Parallel Interface (HiPPI)

The Cray Research High Performance Parallel Interface (HiPPI) is an external channel that provides high-speed communications between 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 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 TCP or 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. HiPPI is well suited to the following applications:

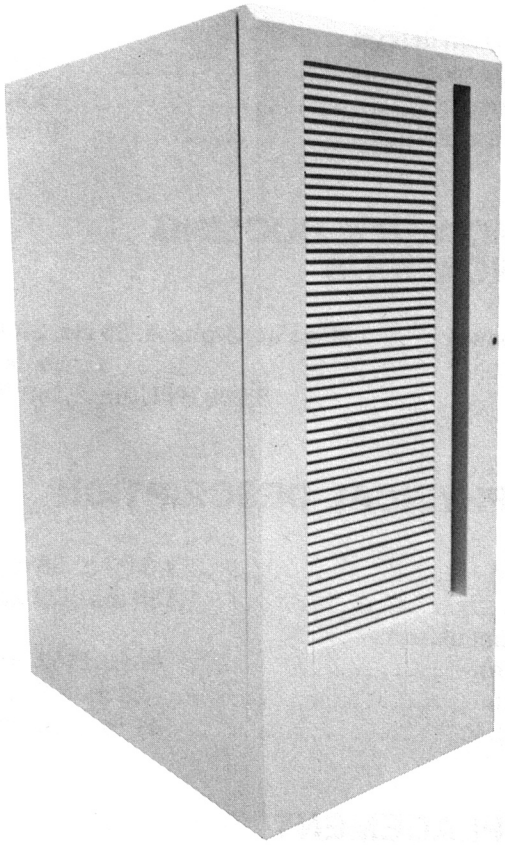
- Distributed applications. The speed of the HiPPI makes more applications suitable for distributed processing. Users can link multiple Cray Research computer systems for maximum supercomputer performance.

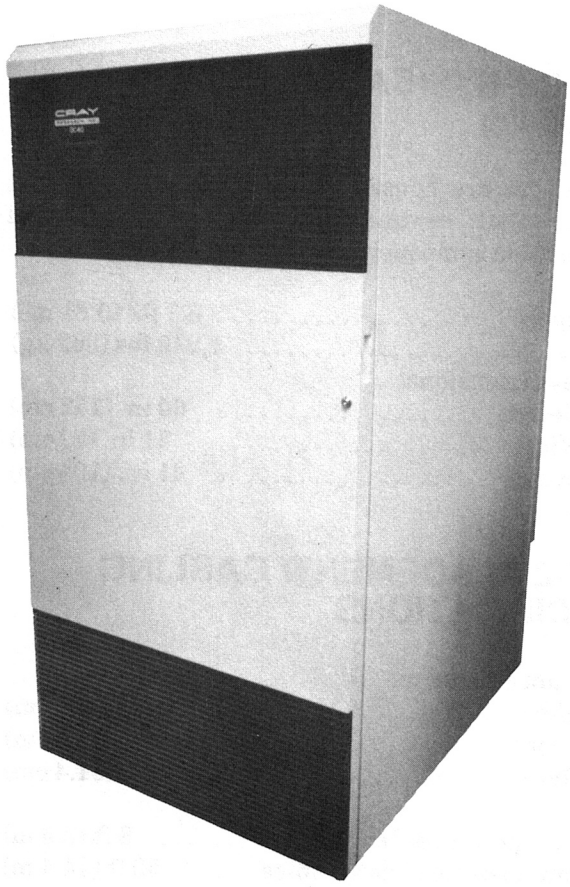
- Raster graphics. Real-time animated graphics are possible when HiPPI is combined with a compatible high-speed frame buffer. Existing devices have delivered up to 60 frames per second on a 512-by-512 raster of 24-bit pixels.

DEC VAX Supercomputer Gateway

Digital Equipment Corporation (DEC) offers a VAX Supercomputer Gateway to enable direct connection between the DEC VAX cluster environment and a CRAY Y-MP computer system.

DS-40 and DS-40D DISK SUBSYSTEMS





CRAY RESEARCH, INC.

Specification Sheet

DS-40 and DS-40D SPECIFICATIONS

DC-40 FEATURES

Sustained transfer rate ..... 9.6 Mbytes/s
Storage capacity ..... 5,200 Mbytes

DCC-2 POWER & COOLING SPECIFICATIONS (four DC-40s)

Required power .... 208 Vac, 3-phase, 60 Hz, 60 A
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 PHYSICAL DESCRIPTION (four DC-40s)

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

Floor space ..... 8.7 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 PLACEMENT/ CABLING SPECIFICATIONS

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

DD-40 FEATURES

Sustained transfer rate ..... 9.6 Mbytes/s
Total data sectors 1,293,216
Total data words 662,126,592

Typical position delays:
- Single track ..... 4 ms
- Average ..... 16 ms
- Full stroke ..... 30 ms

DD-40 POWER & COOLING SPECIFICATIONS

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/ CABLING SPECIFICATIONS

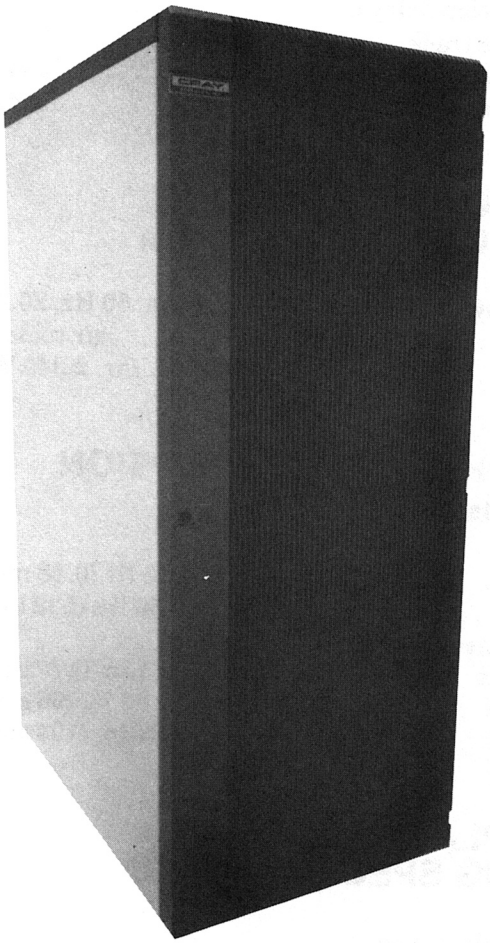
Minimum clearance:
- Sides ..... 1 in. (2.5 cm)
- Front ..... 36 in. (91.4 cm)
- Back ..... 30 in. (76.2 cm)

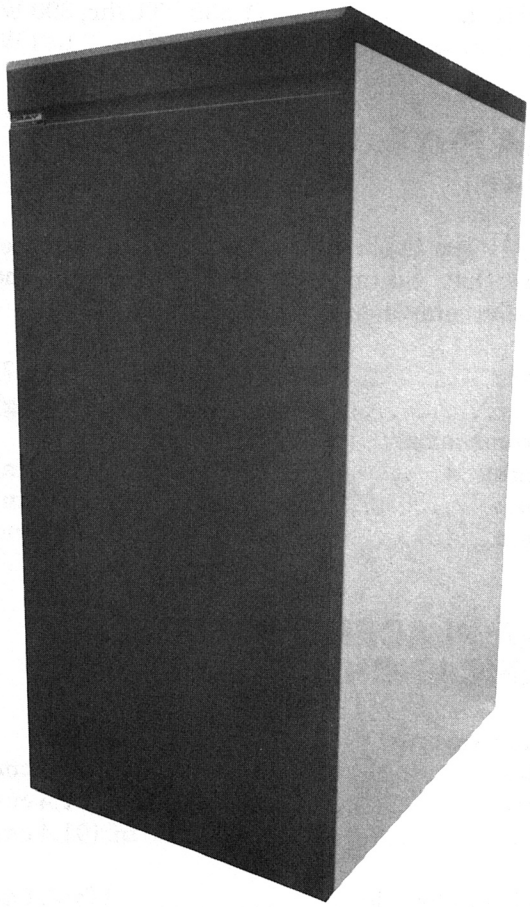
Length of power cable ..... 6 ft (1.8 m)
Maximum length of data cables ..... 20 ft (6 m)

Four Disk 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 since the data path is set at 9.6 Mbytes/s.

DS-41, DS-41D, and DS-41R DISK SUBSYSTEMS





CRAY RESEARCH, INC.

Specification Sheet

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 & COOLING SPECIFICATIONS (four DC-41s)

Required power	208 Vac, 3-phase, 60 Hz, 60 A
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 ft² (0.81 m²)
Weight	1,270 lbs (576 kg)
Cabinet dimensions:	
• Height	67 in. (170 cm)
• Width	31 in. (79 cm)
• Depth	41 in. (104 cm)

DCC-2A PLACEMENT/ CABLING SPECIFICATIONS

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 & COOLING SPECIFICATIONS (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 ft² (0.68 m²)
Weight	1,150 lbs (522 kg)
Cabinet dimensions:	
• Height	67 in. (170 cm)
• Width	26 in. (66 cm)
• Depth	41 in. (104 cm)

DE-41 PLACEMENT/ CABLING SPECIFICATIONS

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 Storage Units (DSUs) are connected to the DCC-2A chassis for the DS-41 Disk Subsystem. Eight DSUs are connected to the DCC-2A chassis for the DS-41D 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 19.2 Gbytes to 38.4 Gbytes; daisy chaining does not increase the 9.6 Mbyte/s transfer rate.

DD-49 DISK DRIVE



CRAY RESEARCH, INC.

Specification Sheet

DD-49 SPECIFICATIONS

DD-49 FEATURES

Storage capacity ..... 1,200 Mbytes

Transfer rate:

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

Total data sectors 297, 696
Total data words 152, 420, 352

Typical position delays:

- Single track ..... 2 ms
- Average ..... 16 ms
- Full stroke ..... 30 ms

POWER & COOLING SPECIFICATIONS

Required power .... 208 Vac, 3-phase, 60 Hz, 20A
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 kgs)

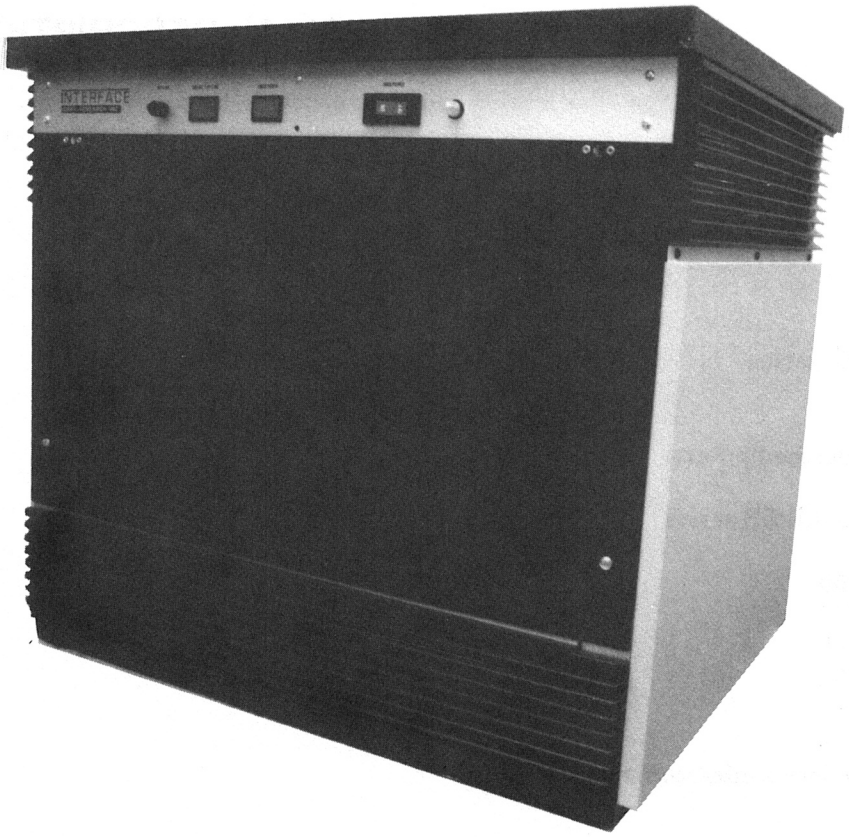
PLACEMENT/CABLING SPECIFICATIONS

Minimum clearance:

- Sides ..... 12 in. (30 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



CRAY RESEARCH, INC.

Specification Sheet

FEI SPECIFICATIONS

FEI FEATURES

Cray Research, Inc. offers hardware interfaces and station software to connect the CRAY Y-MP 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)

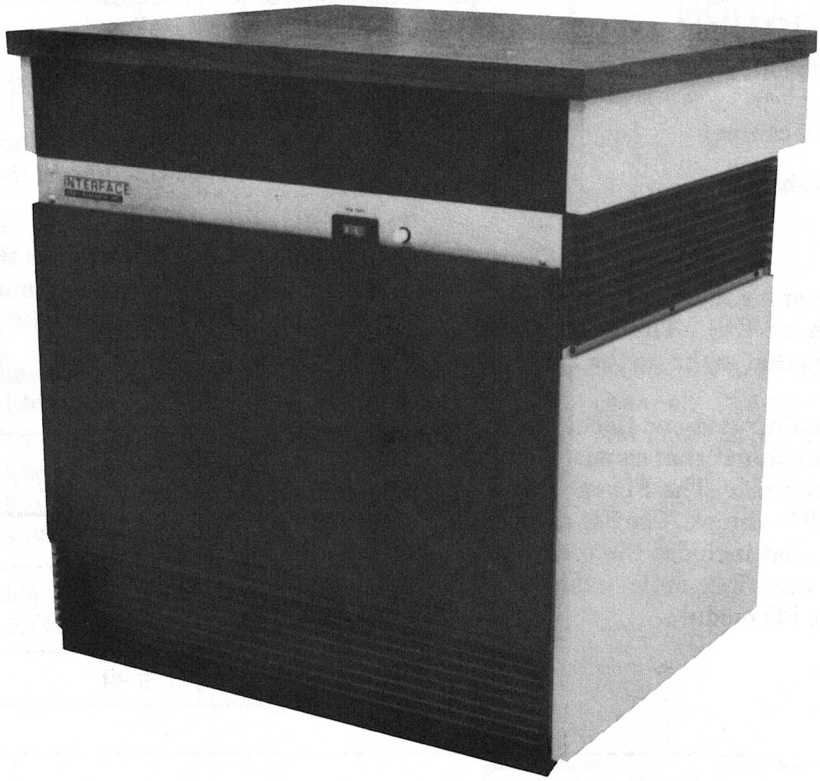
Operating systems:

- 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.

FOL-3 FIBER-OPTIC LINK



CRAY RESEARCH, INC.

Specification Sheet

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 CONFIGURATION

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 dotted line encircles the components that make up 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 FO module.

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

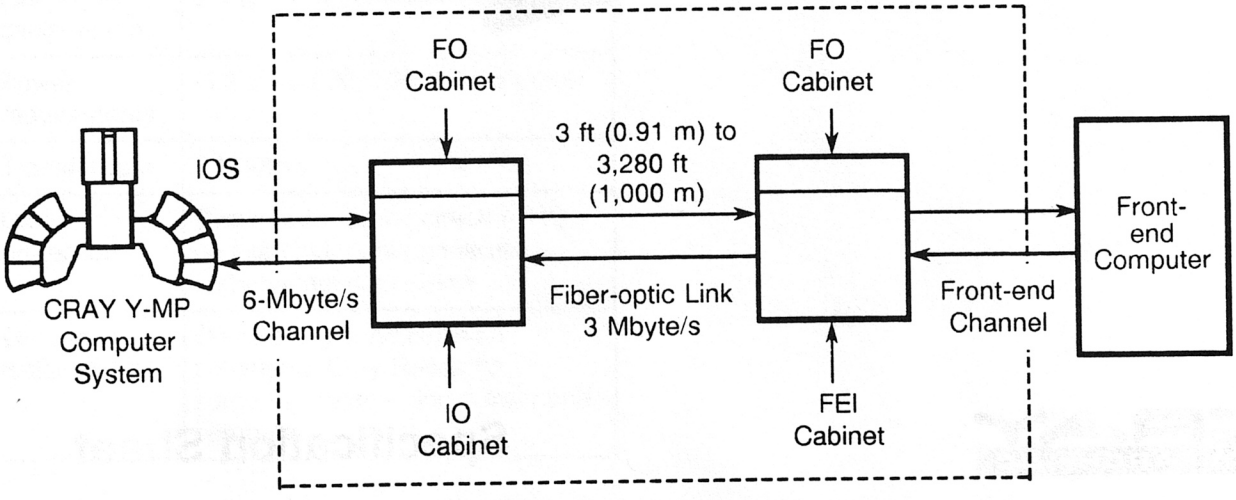
At the FEI mainframe end (remote end of the FOL) of this link is an FEI cabinet. This cabinet consists of an FO cabinet positioned on top of an FEI cabinet. The FEI cabinet contains the modules necessary to communicate with the front-end computer system and a Cray Research IO module. The FO cabinet is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

The electrical kit contains a Cray 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.

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

FOL-3 Equipment List

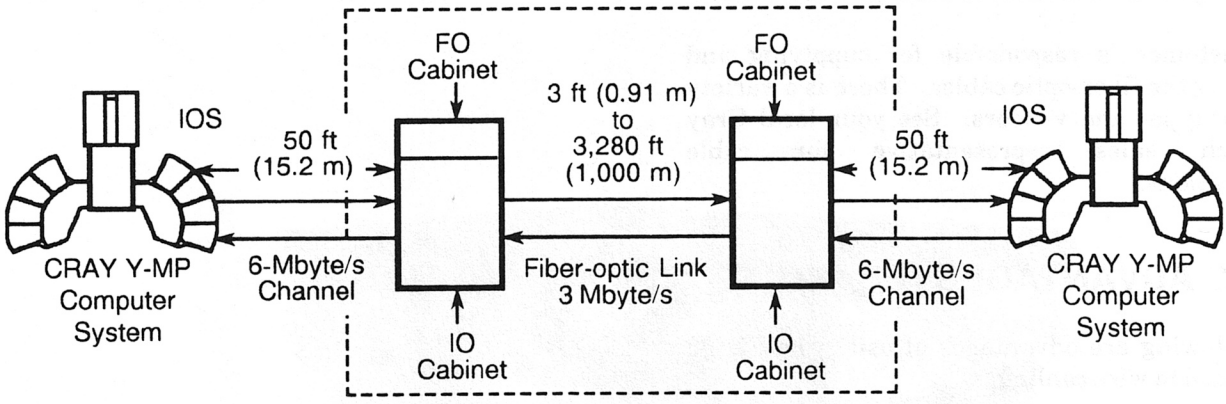
Equipment	Quantity Needed
Electrical kit	One kit per FOL-3
FO cabinet	Two kits for initial installation; one kit per FOL-3 thereafter
IO cabinet	One kit



General FOL-3 Configuration

FOL-3 SPECIFICATIONS

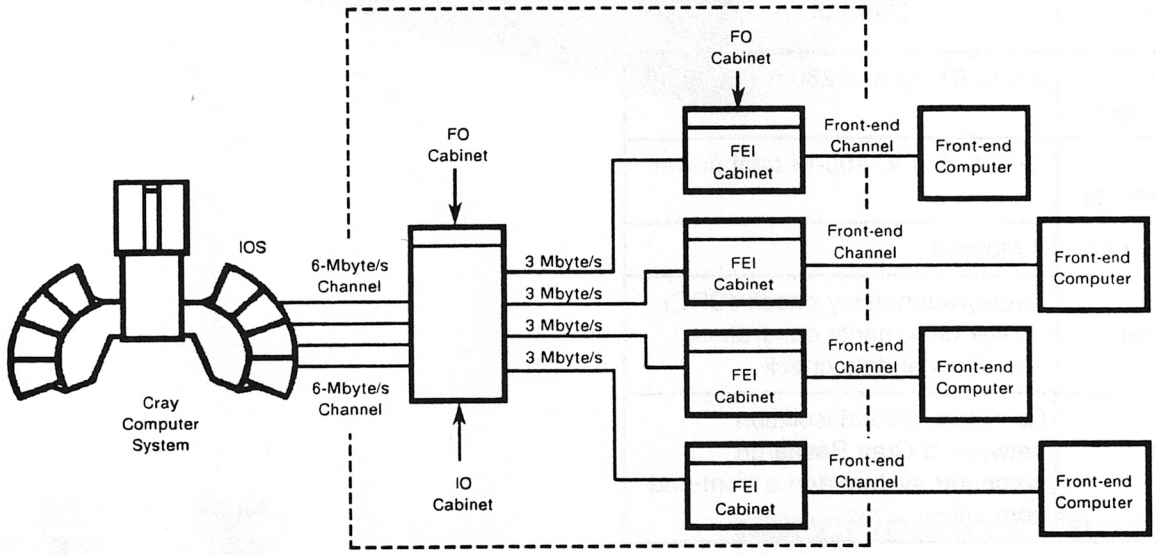
The illustration below shows the general configuration of the FOL-3 when two CRAY Y-MP systems are configured together. The dotted line encircles the components that make up the FOL-3.



FOL-3 Connection between Two CRAY Y-MP Computer Systems

The illustration below shows the FOL-3 connected to a CRAY Y-MP 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 front-end interfaces (FEIs).

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



FOL-3 Configured with a CRAY Y-MP Computer System

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. There is a variety of cable types and vendors. See your local Cray Research sales representative for cable specifications.

FOL-3 ADVANTAGES

The following 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 following table describes FOL-3 features.

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 Mbyte/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

PHYSICAL DESCRIPTION

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

CONTENTS

6 - SOFTWARE OVERVIEW 6-1
Operating Systems 6-1
UNICOS 6-1
COS 6-2
Multiprocessing 6-2
Macrotasking 6-2
Microtasking 6-3
Autotasking 6-3
Fortran Compilers 6-3
CFT77 6-3
CFT 6-4
C Compiler 6-4
Pascal 6-4
Cray Assembler 6-5
Subroutine Libraries 6-5
Utilities 6-5
I/O Subsystem Software 6-6
Communications Software 6-6
Applications 6-7
Software Publications 6-7
UNICOS Operating System 6-8
COS Operating System 6-8
Fortran 6-8
C 6-8
Pascal 6-8
Cray Assembler 6-9
Libraries 6-9
Utilities 6-9
Communications Software 6-9
Applications 6-9
Software Training 6-10

6 - SOFTWARE OVERVIEW

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

The CRAY Y-MP computer system is supported by communications software such as the TCP/IP protocol (a widely-accepted protocol for interconnecting UNIX systems) and Cray proprietary station products for connecting other vendors' systems and workstations to Cray Research computers.

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

OPERATING SYSTEMS

The CRAY Y-MP computer system comes with the UNICOS operating system. The UNICOS operating system is derived from the AT&T UNIX System V operating system. UNICOS is also based in part on the Fourth Berkeley Software Distribution (BSD).

The Cray operating system COS is also available on the CRAY Y-MP computer system. COS provides compatibility with software written for the CRAY X-MP computer system. The operating systems are described in the following subsections.

UNICOS

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

UNICOS 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.

UNICOS offers a large set of utility programs that allows the user to interact with the operating system. In addition to the utilities, UNICOS provides a number of products specifically designed for Cray computer systems. UNICOS supports optimizing, vectorizing, concurrentizing Fortran compilers; an optimizing, vectorizing Pascal compiler; and an optimizing, vectorizing C compiler.

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

COS

The Cray operating system COS is supported on the CRAY Y-MP computer system to provide compatibility with software that utilizes COS written for the CRAY X-MP computer system. COS is a multiprogramming, multiprocessing, and multitasking operating system. It offers a batch environment to the user and supports interactive jobs and data transfers where these are available through the front-end system. COS programs developed in the CRAY X-MP environment can run in a maximum of four processors using up to 16 Mwords of Central Memory on a CRAY Y-MP system.

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

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

MULTIPROCESSING

Multiprocessing partitions an application program into independent tasks and runs them in parallel on the CRAY Y-MP computer system. Multiprocessing results in substantial throughput improvement over serially executed programs. Three multiprocessing methods are available, and they can all work together in a single program. The three methods are macrotasking, microtasking, and autotasking.

Macrotasking

Macrotasking is an implementation of multiprocessing that allows parallel execution of code at the subroutine level on multiple processors. Macrotasking is best suited to programs with larger, longer-running tasks. The user interface to the CRAY Y-MP 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 products.

Microtasking

Microtasking is a multiprocessing technique that allows parallel execution of very small segments of code on multiple processors. An example of this would be individual iterations of DO loops. With microtasking, the programmer can 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

Autotasking is automatic multiprocessing. Autotasking allows user programs to be automatically partitioned over multiple CPUs (without user intervention). 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 CPUs available, both large and small granularity parallelism, and so on. In addition to being fully automatic, autotasking exceeds microtasking in overall performance and in the various levels of parallelism that can be employed.

FORTRAN COMPILERS

The CRAY Y-MP computer system offers the Cray Fortran compiler CFT77 and the Cray Fortran compiler CFT. Both compilers are fully compliant with the ANSI 78 (Fortran 77) standards and offer a high degree of automatic scalar and vector optimization. Both permit maximum portability of programs between different Cray systems and accept many nonstandard 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 CRAY Y-MP system architecture.

CFT77

The Cray Fortran compiler CFT77 is a multipass, optimizing, transportable compiler that processes existing standard Fortran programs. CFT77 uses two basic techniques to improve the execution time of a Fortran program: vectorization and scalar optimization.

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

CFT77 is portable on several levels. Because it is in compliance with the 1978 ANSI standard, programs written for other computer systems have maximum portability to the CRAY Y-MP system with minimum effort. Also, the compiler is designed to run on all

Cray systems so a Fortran program that compiles and runs on one Cray system will run on all Cray systems. Changing from CFT to CFT77 is also easy. In general, programs that compile and execute correctly with the CFT compiler also compile and execute correctly with CFT77.

CFT

The Cray Fortran compiler CFT is supported on the CRAY Y-MP computer system to make it compatible with software written for the CRAY X-MP computer system. CFT compiles Fortran programs that most efficiently implement the common hardware features of the CRAY X-MP and CRAY Y-MP computer systems. CRAY X-MP and CRAY Y-MP compatibility is achieved for up to 16-Mword programs by linking the compiler-produced binaries with Cray-provided, runtime libraries.

CFT automatically generates vectorized machine language code; thus, the power of a vector computer becomes immediately accessible to the user having no prior experience with vectorization techniques. CFT analyzes the innermost loops of a Fortran program to detect vectorizable sequences. The vectorization of inner loops in Fortran programs allows these programs to take advantage of the great speed of vector operations.

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

C COMPILER

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

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

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

PASCAL

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

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

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

CRAY ASSEMBLER

The Cray assembler, CAL, enables a user to closely tailor a program to the architecture of the CRAY Y-MP computer system. Through CAL, a programmer may symbolically express all hardware functions of the Cray system. CAL allows the production of highly efficient, machine language programs. The user may designate program and data information to enable complete control of the mainframe CPUs. This facilitates the full use of various CRAY Y-MP 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 macros for subroutine entry and exit, allowing for easy subroutine linkage.

SUBROUTINE LIBRARIES

Cray software includes subroutines that are callable from CFT77, CFT, C, Pascal, and CAL. The subroutines are divided into libraries, generally on a functional basis. Libraries containing 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 the CRAY Y-MP computer system.

The segment loader, SEGLDR, 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 tools are available for postmortem dump analysis and interpretation.

There are a variety of performance aids to assist in analyzing program performance and optimizing programs with a minimum of effort. These aids include both static and dynamic analyzers, as well as profilers for CPU and I/O usage.

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, offer versatility for users wishing to create and maintain text files. Other system utilities provide for proper management of the system resources.

I/O SUBSYSTEM SOFTWARE

The I/O Subsystem (IOS) software is written in the I/O Processor (IOP) macro assembly language APML. The APML assembler executes in the CPU under control of the operating system to generate IOS object code.

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

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

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

COMMUNICATIONS SOFTWARE

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

The Transport Control Protocol/Internet Protocol (TCP/IP) product allows the CRAY Y-MP computer system to function as a peer in TCP/IP-supported, open networking environments. TCP/IP is a set of computer networking protocols that allow two or more hosts to communicate. Further, it is a set of procedures that allow 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 UNICOS.

Cray Research station software products provide CRAY Y-MP systems 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 mainframe and the Apollo workstation network, DOMAIN.
- CYBER station; joins the Cray system to the Control Data Corporation CYBER 180 series, 70/170, or 700/800 systems to form a powerful computing combination.
- VAX/VMS station; controls the hardware and software link between a DEC VAX Computer System and a Cray computer system.
- MVS station; provides the software connection between an IBM System/370, Extended Architecture, or compatible computer system and a Cray 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 computer system.
- UNIX station; provides Cray operating system access to installations whose front ends run UNIX.

APPLICATIONS

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

The availability of applications for Cray systems is driven largely by customer requirements that are communicated to the software vendors. Cray Research, Inc. provides support for the ongoing process of converting and maintaining application software.

SOFTWARE PUBLICATIONS

A partial list of Cray Research, Inc. software publications appears below. The manuals provide additional information about the software described in this section. These manuals and other user publications can be ordered through Cray Research, Inc. 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
SR-2001	UNICOS User Commands Reference Manual
SR-2012	UNICOS System Calls Reference Manual
SR-2014	UNICOS File Formats and Special Files Reference Manual
SR-2022	UNICOS Administrator Commands Reference Manual
SG-2005	I/O Subsystem (IOS) Operator's Guide for UNICOS
SG-2017	UNICOS Source Code Control System (SCCS) User's Guide
SG-2018	UNICOS System Administrator Guide for CRAY X-MP and CRAY-1 Computer Systems
SG-2050	UNICOS Text Editor Primer
SG-2052	UNICOS Overview for Users

COS Operating System

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

Fortran

Publication Number	Title
SR-0009	Fortran (CFT) Reference Manual
SR-0018	CFT77 Reference Manual

C

Publication Number	Title
SR-2024	Cray C Reference Manual

Pascal

Publication Number	Title
SR-0060	Pascal Reference Manual

Cray Assembler

Publication Number	Title
SR-2003	CAL Assembler Version 2 Reference Manual

Libraries

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

Utilities

Publication Number	Title
SR-0010	Software Tools Reference Manual
SR-0066	Segment Loader (SEGLDR) Reference Manual
SR-0112	Symbolic Debugging Package Reference Manual
SR-0222	CRAY X-MP Multitasking Programmer's Manual

Communications Software

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

Applications

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

SOFTWARE TRAINING

Cray Research, Inc. offers complete training on the software available for the CRAY Y-MP computer system. 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 computer system. More information regarding courses and schedules can be obtained through your local or regional Cray Research, Inc. sales office.

GLOSSARY

GLOSSARY

A

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

Autotasking - Autotasking is automatic multiprocessing; it allows user programs to be automatically partitioned over multiple CPUs without user interface.

B

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

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

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

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

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

C

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

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

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

Central memory - Memory in the CRAY Y-MP mainframe shared by all CPUs.

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

C (continued)

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

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

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

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

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

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

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

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

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

CRI - Cray Research, Inc.

D

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

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

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

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

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

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

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

D (continued)

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

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

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

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

E

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

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

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

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

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

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

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

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

EX - Normal Exit (instruction). The EX instruction initiates an Exchange Sequence.

Exchange mechanism - The technique used in the CRAY Y-MP 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.

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

F

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

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

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

FOL-3 - Fiber-optic link. The CRI 3-Mbyte/s fiber-optic link allows an FEI to be separated from a Cray 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 F register in the Exchange Package contains the FPE flag. The FPE flag sets when a Floating-point Range error occurs in any of the floating-point functional units and the Interrupt-on-floating-point Error (IFP) flag is set.

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

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

G

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

H

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

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

HSX - High-speed External channel. The HSX channel is a high-speed, full-duplex, external channel capable of transferring data at a maximum rate of 100 Mbytes per second.

HiPPI - High Performance Parallel Interface. The HiPPI channel is an external channel that provides high-speed communications between the IOS and peripheral equipment.

I

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

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

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

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

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

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

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

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

IOC - I/O Subsystem Chassis.

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

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

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

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

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

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

J

J - The unconditional branch instruction.

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

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

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

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

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

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

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

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

L

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

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

M

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

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

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

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

MGS - Motor Generator Set. The MGS receives 460-Vac, 60-Hz or 380-Vac, 50-Hz power and converts it to 208-Vac, 400-Hz power for supply voltage to the computer system logic. The MGS also isolates system power from transients and fluctuations on the commercial power lines and provides an override of $\frac{1}{2}$ second in case of intermittent power outages.

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

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

M (continued)

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

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

N

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

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

O

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

P

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

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

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

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

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

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

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

P (continued)

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

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

Q

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

R

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

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

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

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

S

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

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

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

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

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

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

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

S (continued)

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

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

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

T

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

U

UNICOS - An operating system for Cray computer systems based primarily on the AT&T UNIX System V and partially on the Fourth Berkeley Software Distribution (BSD). UNICOS is essentially the same in philosophy, structure, and function as UNIX, but has been enhanced to exploit the power of Cray computer systems.

V

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

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

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

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

W

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

X

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

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

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

Y

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

Z

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

INDEX

INDEX

1-parcel instruction format
    with combined j and k fields, 2-38
    with discrete j and k fields, 2-37
100-Mbyte/s channel, 1-4, 3-3, 4-1
1000-Mbyte/s channel, 1-4, 2-25, 4-1
2-parcel instruction format
    with combined i, j, k, and m fields, 2-39
    with combined j, k, and m fields, 2-38
22-bit immediate constant, 2-38
24-bit immediate constant, 2-40
24-bit integer arithmetic, 2-10, 2-53
3-parcel instruction format with combined
  m and n fields, 2-40
32-bit integer arithmetic, 2-10, 2-53
6-Mbyte/s channels, 1-6, 2-25, 3-3, 3-4, 4-1, 5-6
64-bit integer arithmetic, 2-10, 2-53

A registers, *See* Address registers
Addition algorithm, 2-16
Address Add functional unit, 2-7, 2-44
Address functional units, 2-7, 2-44
Address Multiply functional unit, 2-7, 2-44
Address processing, 1-3, 2-3, 2-6
Address registers, 2-5, 2-45, 2-47
Algorithm
    addition, 2-16
    division, 2-18
    multiplication, 2-17
AND function, 2-10, 2-56, 2-57
Applications programs, 6-7, 6-9
Assembler, 6-5
Autotasking, 6-2, 6-3
Auxiliary I/O processor (XIOP), 3-3, 5-7

B registers, *See* Intermediate registers
Bidirectional Memory
    mode flag, 2-25
    mode, 2-1, 2-28
    transfers, 2-1, 2-50
Bit count instructions, 2-60
    scalar leading zero count, 2-61
    scalar population count, 2-60
    scalar population count parity, 2-61
    vector population count, 2-61
    vector population count parity, 2-61
Branch instructions, 2-37, 2-61
    conditional, 2-62
    error exit, 2-62
    normal exit, 2-62
    return jump, 2-62
    unconditional, 2-61
Buffer I/O processor (BIOP), 3-3, 5-1
Buffer Memory, 1-3, 3-1, 3-3, 3-4
Buffers, Instruction, 2-27

C language, 6-1, 6-4, 6-5
CA register, *See* Current Address register
Central Memory, 1-1, 1-3, 2-1, 2-3, 2-22, 2-27,
  3-1, 3-3, 4-1
Central Processing Unit
    computation section characteristics, 2-3
    instruction format, 2-37
    instructions, 2-37
    shared resources, 2-1
CFT Fortran compiler, 6-1, 6-3, 6-4, 6-5, 6-8
CFT77 Fortran compiler, 6-1, 6-3, 6-4, 6-5, 6-8
Channel
    100-Mbyte/s, 1-4, 3-3, 4-1
    1000-Mbyte/s, 1-4, 2-25, 4-1
    6-Mbyte/s, 1-6, 2-25, 3-3, 3-4, 4-1, 5-4
    control, 2-63
Channel Limit register (CL), 2-63
Checkbyte, 2-1, 2-65, 4-2
CIP register, *See* Current Instruction
  Parcel register
CL register, *See* Channel Limit register
CLN register, *See* Cluster Number register
Clock
    Programmable, 2-27
    Real-time, 2-2, 2-48, 2-63
Clock period, 2-2, 2-8, 2-27, 2-28, 2-29, 2-30
Cluster Number register (CLN), 2-2, 2-24, 2-28
Communication, interprocessor, 2-2
Compressed index, 2-36
Computation section, 2-3

Conditional branch instructions, 2-62
Configurations of system, 2-68, 2-72, 2-76
Control, interprocessor, 2-2
Conventions, notational, vi
Correctable Memory Error Mode flag, 2-23, 2-25
COS operating system, 6-1, 6-2, 6-6, 6-7, 6-8
CP, *See* clock period
CPU, *See* Central Processing Unit
CPU operating registers
    A registers, 2-5
    address registers, 2-5
    B registers, 2-5
    S registers, 2-6
    scalar registers, 2-6
    T registers, 2-6
    V registers, 2-6
    Vector registers, 2-6
CRAY Y-MP2 mainframe, 1-1, 2-1, 3-1, 2-75
CRAY Y-MP4 mainframe, 1-1, 2-1, 3-1, 2-71
CRAY Y-MP8 mainframe, 1-1, 2-1, 3-1, 2-67
Current Address register (CA), 2-48, 2-63
Current Instruction Parcel register (CIP), 2-27, 2-28

Daisy chain, 5-3, 5-4
Data Base Address register (DBA), 2-23, 2-25
Data formats
    integer, 2-10
    floating-point, 2-13
Data Limit Address register (DLA), 2-24, 2-25
DBA register, *See* Data Base Address register
DCU, *See* disk storage unit
DD-49, 5-5, 5-13
Deadlock, 2-2, 2-24
Disable Floating-point Interrupt, 2-25
Disk control unit, 1-5, 5-1
Disk I/O processor (DIOP), 3-3, 5-1
Disk storage units, 4-1, 5-1, 5-3
Division algorithm, 2-19
DL flag, 2-25, 2-26
DLA register, *See* Data Limit Address register
Double-precision numbers, 2-12, 2-21
Double-shift instructions, 2-60
DS-40, 5-2, 5-3, 5-9
DS-41, 5-3, 5-4, 5-11

Enable floating-point interrupt, 2-26
Error detection and correction, 2-1, 3-2, 4-2
Error exit, 2-62
Error Exit flag, 2-25
Exchange Address (XA) register, 2-24
Exchange mechanism, 2-22
Exchange Package, 2-22
    address base and limit, 2-23
    Memory Error data, 2-23
    Processor Number, 2-22
    Vector Not Used (VNU), 2-26
    Waiting for Semaphore, 2-27
Exchange Package registers
    A registers, 2-27
    Cluster Number register, 2-24
    Exchange Address register, 2-24
    Flag register, 2-25
    Mode register, 2-26
    Program Address register, 2-23
    S registers, 2-27
    Vector Length register, 2-24
Exchange sequence, 2-22, 2-29
Exclusive NOR function, 2-10
Exclusive OR function, 2-10, 2-56
External Interrupts flag, 2-26

F register, *See* Flag register
FEI, 1-5
FEI-1, 5-5
FEI-3, 5-6
Flag register, 2-25
Flags
    Bidirectional Memory Mode, 2-26
    Correctable Memory Error Mode, 2-26
    Enable Second Vector Logical, 2-26
    Extended Addressing mode, 2-26
    External Interrupts, 2-26
    Floating-point Error Mode, 2-26
    Monitor Mode, 2-26
    Operand Range Error , 2-26
    Program Range Error, 2-25
    Program State, 2-26
    Uncorrectable Memory Error Mode, 2-26
Floating-point
    Add functional unit, 2-9, 2-44
    addition, 2-17
    arithmetic, 2-12
    data format, 2-13
    division, 2-19
    Error flag, 2-25, 2-28
    exponent ranges, 2-14
    functional units, 2-9
    Interrupt flag, 2-25, 2-26
    instructions
        addition and subtraction, 2-54
        multiplication, 2-55
        range errors, 2-54

Floating-point instructions (continued)
reciprocal Approximation, 2-56
reciprocal iteration, 2-56
multiplication, 2-18
Multiply functional unit, 2-10, 2-44, 2-53
normalized numbers, 2-15
Range errors, 2-15
Reciprocal Approximation functional unit, 2-9, 2-44
Floating-point arithmetic, 2-12
exponent range, 2-14
Fortran compilers
CFT , 6-1, 6-3, 6-4, 6-5, 6-8
CFT77 , 6-1, 6-3, 6-4, 6-5, 6-8
Functional units, 2-7, 2-44
address, 2-7, 2-44
Address Add, 2-7, 2-44
Address Multiply, 2-7, 2-44
Floating-point, 2-9, 2-44
Floating-point Add, 2-9, 2-44
Floating-point Multiply, 2-9, 2-44
Full Vector Logical, 2-8, 2-44
Reciprocal Approximation, 2-9, 2-44
scalar, 2-7, 2-44
Scalar Add, 2-8, 2-44
Scalar Logical, 2-8, 2-44
Scalar Population/Parity/Leading Zero, 2-8, 2-44
Scalar Shift, 2-8, 2-44
Second Vector Logical, 2-9, 2-26, 2-44
vector, 2-8, 2-30, 2-44
Vector Add, 2-8, 2-44
Vector Population/Parity, 2-9, 2-44
Vector Shift, 2-8, 2-44,
Functional instruction summary, 2-45
Functional units instruction summary, 2-44

g field, 2-37, 2-38, 2-39, 2-40
Gather instruction, 2-34
General instruction form, 2-37

h field, 2-37, 2-38, 2-39, 2-40
Heat Exchanger Unit, 1-6, 1-7
HiPPI, 5-7
HSX, 1-4, 3-3, 5-5

i field, 2-37, 2-38, 2-39, 2-40
ICP flag, 2-25, 2-26
IBA register, *See* Instruction Base Address register
ILA register, *See* Instruction Limit Address register
Inclusive OR function, 2-10
Instruction
bit count, 2-60
scalar leading zero count, 2-61
scalar population count, 2-60
scalar population count parity, 2-61
vector population count, 2-61
vector population count parity, 2-61
branch, 2-37, 2-61
conditional, 2-62
error exit, 2-62
normal exit, 2-62
return jump, 2-62
unconditional, 2-61
buffers, 2-27
functional summary, 2-45
functional unit summary, 2-44
general form, 2-37
monitor mode
channel control, 2-63
cluster number, 2-64
Interprocessor interrupt, 2-64
Operand Range Error interrupt, 2-64
Performance counters, 2-65
Programmable Clock Interrupt, 2-64
set Real-time Clock, 2-63
shift, 2-59
summary, 2-44, 2-45
syntax,
format, 2-37
monitor mode, 2-43
special register values, 2-42
vector, 2-32
merge, 2-59
Instruction Base Address register, 2-23, 2-25
Instruction buffers, 2-27
Instruction fetch, 2-27
Instruction format
1-parcel with combined j and k fields, 2-38
1-parcel with discrete j and k fields, 2-37
2-parcel with combined i, j, k, and m fields, 2-39
2-parcel with combined j, k, and m fields, 2-38
3-parcel with combined m and n fields, 2-40
Instruction Limit Address register (ILA), 2-23, 2-25
Instruction issue, 1-3, 2-22, 2-27
Instruction parcel, 2-27, 2-37, 2-39, 2-40
Instructions, general form for, 2-37

Integer arithmetic, 2-10, 2-52
Integer data formats, 2-11
Inter-register transfers, 2-47
    A registers, 2-47
    S registers, 2-48
    Semaphore registers, 2-49
    V registers, 2-49
    Vector Length register, 2-49
    Vector Mask register, 2-49
Intermediate registers, 2-5
    B registers, 2-5, 2-6, 2-22, 2-38, 2-40, 2-50, 2-51, 2-61
    T registers, 2-5, 2-6, 2-22, 2-38, 2-50, 2-51
Internal CPU interrupt request, 2-25
Interprocessor interrupt instructions, 2-64
I/O Chassis (IOC), 1-1, 1-3, 1-4, 3-1, 3-4, 4-1, 5-1
I/O Processor, 1-3, 1-4, 1-5, 1-6, 3-1, 3-2, 3-5, 5-1, 5-2, 5-3, 6-6
I/O Subsystem, 1-2, 1-3, 1-4, 1-5, 1-6, 1-8, 1-9, 2-2, 3-1,3-2, 3-3, 3-4, 3-5, 3-6
Issue, 1-3, 2-22, 2-26, 2-27

j field, 2-37, 2-38, 2-39, 2-40

k field, 2-37, 2-38, 2-39, 2-40

Loads, 2-51
Local Memory, 3-1, 3-2, 3-3, 5-2
Logical
    AND function, 2-10
    differences, 2-10
    exclusive NOR function, 2-10
    exclusive OR function, 2-10
    inclusive OR function, 2-10
    instructions
        differences, 2-57
        equivalence, 2-58
        merge, 2-59
        products, 2-57
        sums, 2-57
        vector mask, 2-58
Lower Instruction Parcel register (LIP), 2-28
Lower Instruction Parcel-1 register (LIP1), 2-28

m field, 2-37, 2-38, 2-39, 2-40
M register, *See* Mode register
Macrotasking, 6-2
Master I/O processor (MIOP), 2-25, 3-3, 5-7
Memory, *See* Buffer Memory, Central Memory, or Local Memory
Memory Error data fields, 2-23
Memory transfer instructions
    bidirectional, 2-50
    loads, 2-51
    references, 2-50
    stores, 2-50
Merge, 2-5, 2-6, 2-9, 2-10, 2-35, 2-56, 2-59
Microtasking, 6-3
Mode register (M), 2-26
Monitor instructions, 2-43, 2-63
    channel control, 2-63
    Cluster number, 2-64
    Interprocessor Interrupt, 2-64
    Operand Range Error interrupt, 2-64
    Performance counters, 2-65
    Programmable Clock Interrupt, 2-64
    set Real-time clock, 2-63
Monitor mode
    flag, 2-26
    instructions, 2-63
Motor-generator sets, 1-2, 1-6, 1-8
Multiplication algorithm, 2-17
Multiprocessing, 6-2
Multitasking, 1-1, 2-2, 2-26, 2-28, 6-2, 6-4, 6-9

n field, 2-37, 2-40, 2-41
Network interfaces, 1-5, 3-4, 5-1, 5-5
Newton's method, 2-19
Next Instruction Parcel register (NIP), 2-27
Normal Exit flag, 2-25
Normal exit, 2-62
Normalized floating-point numbers, 2-15
Notation conventions, vi

Operand
    Range Error Interrupt instructions, 2-64
    Range Error flag, 2-24, 2-25
    Range Error Mode flag, 2-24, 2-25
Operating registers, *See* CPU operating registers
Operating systems
    COS, 6-1, 6-2, 6-6, 6-7, 6-8
    UNICOS , 6-1, 6-2, 6-4, 6-6, 6-7, 6-8
Operator Workstation, 1-6, 3-4, 5-6

P register, *See* Program Address register
Parcel address, 2-61

Parcels, 2-37
Parity, *See* Register Parity
Pascal, 6-1, 6-4, 6-5, 6-8
Performance monitor, 2-28
Pipelining, 2-29,2-30, 2-31
PN, *See* Processor number
Population count
    scalar, 2-60
    scalar parity, 2-61
    vector, 2-61
    vector parity, 2-61
Power distribution units, 1-2, 1-6, 1-9
Processor Number (PN), 2-22
Program
    Address register (P), 2-23, 2-27
    Range error, 2-24, 2-25
    Range Error flag, 2-24, 2-25
    State register (PS), 2-26, 2-28
Programmable clock, 2-28
Programmable clock interrupt instructions, 2-64
Programming languages
    C, 6-1, 6-4, 6-5
    CFT, 6-1, 6-3, 6-4, 6-5, 6-8
    CFT77, 6-1, 6-3, 6-4, 6-5, 6-8
    Pascal, 6-1, 6-4, 6-5, 6-8

Read address bank, 2-23
Read mode, 2-23
Real-time Clock register (RTC), 2-2, 2-48, 2-63
Reciprocal Approximation functional unit, 2-9, 2-44
Reciprocal Approximation functional unit iterations, 2-19, 2-20
Register entry instructions
    A registers, 2-45
    S registers, 2-46
    V registers, 2-47
    Semaphore registers, 2-47
Register parity, 2-6, 2-24, 2-25, 2-26
Registers
    Address (A), 2-5, 2-45, 2-47
    Cluster Number (CLN), 2-2, 2-24, 2-28
    Current Instruction Parcel (CIP), 2-28
    Data Base Address, 2-24
    Data Limit Address, 2-24
    Exchange Address (XA), 2-24
    Exchange, *See* Exchange registers
    Flag (F), 2-25
    Instruction Base Address, 2-23
    Instruction Limit Address, 2-23
    Intermediate
        B registers, 2-5, 2-6, 2-22, 2-38, 2-40, 2-50, 2-51, 2-61
        T registers, 2-5, 2-6, 2-22, 2-38, 2-50, 2-51
    Lower Instruction Parcel (LIP), 2-27
    Mode (M), 2-25
    Next Instruction Parcel (NIP), 2-27
    operating, *See* CPU operating registers
    Program Address, 2-27
    Program State (PS), 2-26, 2-28
    Real-time Clock register, 2-2, 2-48, 2-63
    Scalar registers (S), 2-6, 2-46, 2-48
    Semaphore, 2-2
    shared, 2-2
    Shared Address, 2-2
    Shared Scalar, 2-2
    Vector Length, 2-6, 2-33, 2-35
    Vector Mask, 2-6, 2-22, 2-33, 2-35
Return jump, 2-62
RTC register, *See* Real-time Clock register

S registers, *See* Scalar registers
SB registers, *See* Shared Address registers
Scalar
    Add functional unit, 2-7, 2-44
    functional units, 2-7, 2-44
    Logical functional unit, 2-7, 2-44
    registers (S), 2-6, 2-46, 2-48
    Population/Parity/Leading Zero functional unit 2-7, 2-44
    processing, 1-1, 1-3, 2-3
    Shift functional unit, 2-7, 2-44
Scatter instruction, 2-34
SECDED, 2-1, 2-22, 3-4, 4-2
Second Vector Logical unit enable/disable, 2-25
Segmentation, 2-28, 2-29, 2-31
Semaphore registers, 2-2, 2-22
Shared
    address registers, 2-2, 2-22
    registers, 2-2
    resources of CPU, 2-1
    scalar registers, 2-2, 2-22
Shift instructions, 2-59
SM registers, *See* Semaphore registers
Software overview, 6-1
Solid-state Storage Device, 1-2, 1-4, 1-6, 1-8, 1-9, 4-1, 4-2, 4-3, 4-4
Special CAL Syntax, 2-43
Special register values, 2-42
ST registers, *See* Shared Scalar registers
Status register, 2-28

Stores, 2-50
Symbolic notation,
general syntax, 2-37
special syntax form, 2-43
Syndrome, 2-23

T registers, *See* Intermediate scalar registers
Twos complement integer arithmetic, 2-3, 2-7,
Twos complement integer arithmetic,
(continued) 2-8, 2-10, 2-11, 2-49, 2-53

UNICOS, 6-1, 6-2, 6-4, 6-6, 6-7, 6-8
Unconditional branch instruction, 2-61
Uncorrectable Memory Error Mode flag, 2-23,
2-25, 2-28
Unnormalized floating-point value, 2-9, 2-14,
2-15, 2-48, 2-53
Utilities, 6-4, 6-5, 6-9

V registers, *See* Vector registers
Vector
Add functional unit, 2-8, 2-44
chaining, 2-31
instructions, 2-32
Length register, 2-6, 2-33, 2-35
Logical functional units, 2-8, 2-9, 2-44
Mask register, 2-6, 2-22, 2-33, 2-35
merge instruction, 2-59
Population/Parity functional unit, 2-8,
2-44
processing, 1-1, 1-3, 2-3, 2-30, 2-31
Shift functional unit, 2-8, 2-44
VL register, *See* Vector Length register
VM register, *See* Vector Mask register
VNU - vector not used, 2-26

Word boundary, 2-37
Workstation, 1-6, 3-3, 3-4, 5-6
WS flag, 2-27

XA register, *See* Exchange Address register
X-mode, 2-6, 2-10, 2-11, 2-26, 2-37, 2-40, 2-42,
2-45, 2-53, 2-65

Y-mode, 2-6, 2-10, 2-11, 2-26, 2-37, 2-40, 2-42,
2-45, 2-53, 2-65

Reader Comment Form

Title: **CRAY Y-MP Computer Systems Functional Description Manual** **Number:** **HR-04001-0C**

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

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY CARD

FIRST CLASS PERMIT NO 6184 ST. PAUL, MN

POSTAGE WILL BE PAID BY ADDRESSEE

CRAY RESEARCH, INC.

Attention: HARDWARE PUBLICATIONS
770 Industrial Boulevard
Chippewa Falls, WI 54729

Fold

STAPLE

TRANSMITTAL NOTICE

Date: 6-28-90

Publication Number: HR-04001-0C

Publication Title: CRAY Y-MP Computer Systems Functional Description Manual

Remove	Insert	Page Numbers	# of Pages	General contents of manual described and changes (if any) listed
★		Entire manual	All	The CRAY Y-MP Computer Systems Functional Decription Manual, publication number HR-04001-0B, and change packet HR-04001-0B1 are obsolete.
	★	Entire manual	180	This revision incorporates change packet HR-04001-0B1 and adds information on the new DS-41 Disk Subsystem.

Cray Research, Inc.

CRAY RESEARCH, INC.

DISTRIBUTION
2360 PILOT KNOB ROAD
MENDOTA HEIGHTS, MN 55120

(612) 681-5923 FAX (612) 681-5920

PICKING TICKET

PICKING SLIP # 12

ORDER #	DATE ENTERED	ENTERED BY	STATUS	CUSTOMER P.O. #	SHIP VIA	PAGE
10033182	02/27/91	MD	BE	NONE	GS	

SHIP TO

ENTRY	ITEM #	DESCRIPTION	BIN	U/M	QTY. ORD.	QTY. SHIPPED	QTY. B. O.	VALUE
001	HR-4001 C	CRAY YMP FUNCTIONAL DESC MAN	0006717	EA	1	1		

SHIPPED VIA

- [X] UPS
- [X] UPS BLUE
- [ ] TRUCK
- [ ] EXPRESS (DOM. or INT'L)
- [ ] INTERNATIONAL
- [ ] CHIPPEWA TRUCK

DATE SHIPPED	PKG. WT.	CHARGES	PKG. WT.	CHARGES	TOTAL PKGS. SHIPPED
2/27					
FILLED BY					
CHECKED BY					1

TOTA