Cray Product Familiarization

Overview Presentation

Software Training Department

Cray Research, Inc.
2520 Pilot Knob Road
Mendota Heights, MN 55120
(612) 452-9410

January, 1990

RECORD OF REVISION PUBLICATION NUMBER TR-PROD FAM

Requests for copies of Cray Research publications and comments about these publications should be directed to:

CRAY RESEARCH, INC

2520 Northland Drive
Mendota Heights MN 55120

Revision	Description
A	September, 1986 Original printing
B	September, 1987 Reorganized and updated
C	November, 1987 Reorganized and updated
D	May, 1988 Updated to include Y-MP and X-MP EA
E	December, 1988 Updated to include Cray Ada, Lisp, and new options
F	April, 1989 Updated to include new Y-MP models, UNICOS 5.0 features, and new networking options
G	January, 1990 Updated Cray System Performance Section

Table of Contents

Cray Product Familiarization

Description

This course is designed to give Cray employees and customers a general overview of all the Cray Research hardware and software products. Because of the different needs of the various types of students, this course is presented in the following different versions.

Product Familiarization for Cray Non-technical Employees

This version of the course is for the Cray employee who does not have a strong background in the technical aspects of computers. It is **strongly** recommended that an employee who has little knowledge of computers take the two-day computer fundamentals course before attending this course.

Product Familiarization for New Cray Technical Employees and Cray Customers

This version of the course is intended for the newly hired Cray technical employee and Cray customers. For Cray employees, this course should be taken as soon as possible after the employee has been hired.

Product Familiarization for Cray Sales Employees

This version of the course differs from the others in that the course is presented using product presentation materials that are used in actual sales presentations. The materials are organized differently in order to provide a showcase of available presentation materials.

Product Familiarization for Technical People

This course is for the Cray employee or customer who has been in the computer industry for a while and needs an update on the newer Cray hardware and software products. This course can be tailored to the individual needs of a specific audience. Depending on the audience, the length of this class may vary from several hours to two days.

Cray Product Familiarization
Course Objectives

Upon successful completion of this course, the learner should be able to:

1. Identify the major Cray Research hardware products and explain their main characterlstics and applications

2. Identify the major Cray Research software products and explain their main characteristics and applications

3. Explain how the unique architectural features of the Cray Research hardware contribute to its high performance

4. Explain how the unique features of the software products take advantage of the hardware

5. Identlfy some of the main applIcations for which Cray Products are used.

Cray Product Familiarization
Course Outline

I. Introduction

II. Cray Product Overview

- A. Product History
- B. Hardware Products
- C. Software Products
- D. Product Applications

III. Cray Hardware Products

- A. General CRAY Architecture
- B. CRAY Y-MP Computer Systems
- C. CRAY X-MP EA Computer Systems
- D. CRAY-2 Computer Systems

IV. Cray Software Products

- A. The COS Operating System
- B. The UNICOS Operating System
- C. Communications and Networking
- D. Cray Programming Languages

V. Cray Performance and Applications

- A. Cray Programming Languages
- B. Cray Applications

SECTION 1

Cray Product Familiarization

Product Overview

Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK.

ABOUT THE COMPANY

Cray Research, Inc. was organized in 1972 by Seymour R. Cray, a leading designer of large-scale scientific computers, along with a small group of computer industry associates. The company and its subsidiaries are engaged in the design, development, and manufacture of large-scale, high-speed computing systems and the marketing and support of such systems. **The company's computer systems are used primarily for simulation of physical phenomena.**

Initially, the market for Cray computer systems seemed to lie strictly in weather, energy, and defense research. But, by 1978, because Cray computer systems are **general purpose** in design, it became clear that a commercial market existed that would extend the marketplace beyond the traditional Cray customer. **Commercial applications of the machines have widened the market to include users in petroleum research, fluid dynamics, geophysics and seismic analysis, structural analysis, electrical engineering, aerospace design, computational chemistry, and other disciplines.**

WHAT IS A SUPERCOMPUTER?

Supercomputers, by definition, are the fastest, most powerful computers commercially available at any time. Supercomputers are considered to be an order of magnitude (10 times) faster than the next most powerful class of computers.

WHY A CRAY?

Cray computer systems can solve complex mathematical and logical operations that involve large quantities of data and have the ability to carry out these operations at phenomenal speeds. The speed of each system is partially derived from its unique semi-circular design, which allows short wire lengths to carry the electrical signals. Patented cooling systems solve the heat generation problem that is produced by such advanced computation. In addition, new designs have made multiprocessing cost-effective for Cray users.

On a Cray computer system customers can:

- Solve problems that otherwise could not be solved
- Solve problems which are difficult or impossible to physically measure
- Solve problems more cost effectively than many physical tests or tests done on slower computer systems
- Solve very complex problems within a specific response time
- Achieve high quality results by allowing the investigation of more options

Because of the ever-expanding range of uses of Cray computers, the company has become known as the developer of **"the world's fastest general-purpose computers"**.

CRAY-1 Supercomputer

The CRAY-1 was Cray Research, Inc.'s first supercomputer. Development began in 1972 with Seymour Cray and a small group of developers. The first computer (serial number 1) was delivered to a customer in 1976. Cray-1 systems were a part of the product line until 1984 when the X-MP/1 replaced the CRAY-1M as the low-end model in the product line.

A CRAY-1 is identifiable by its squared seat corners, flat column panels, and aluminum protrusions on its top.

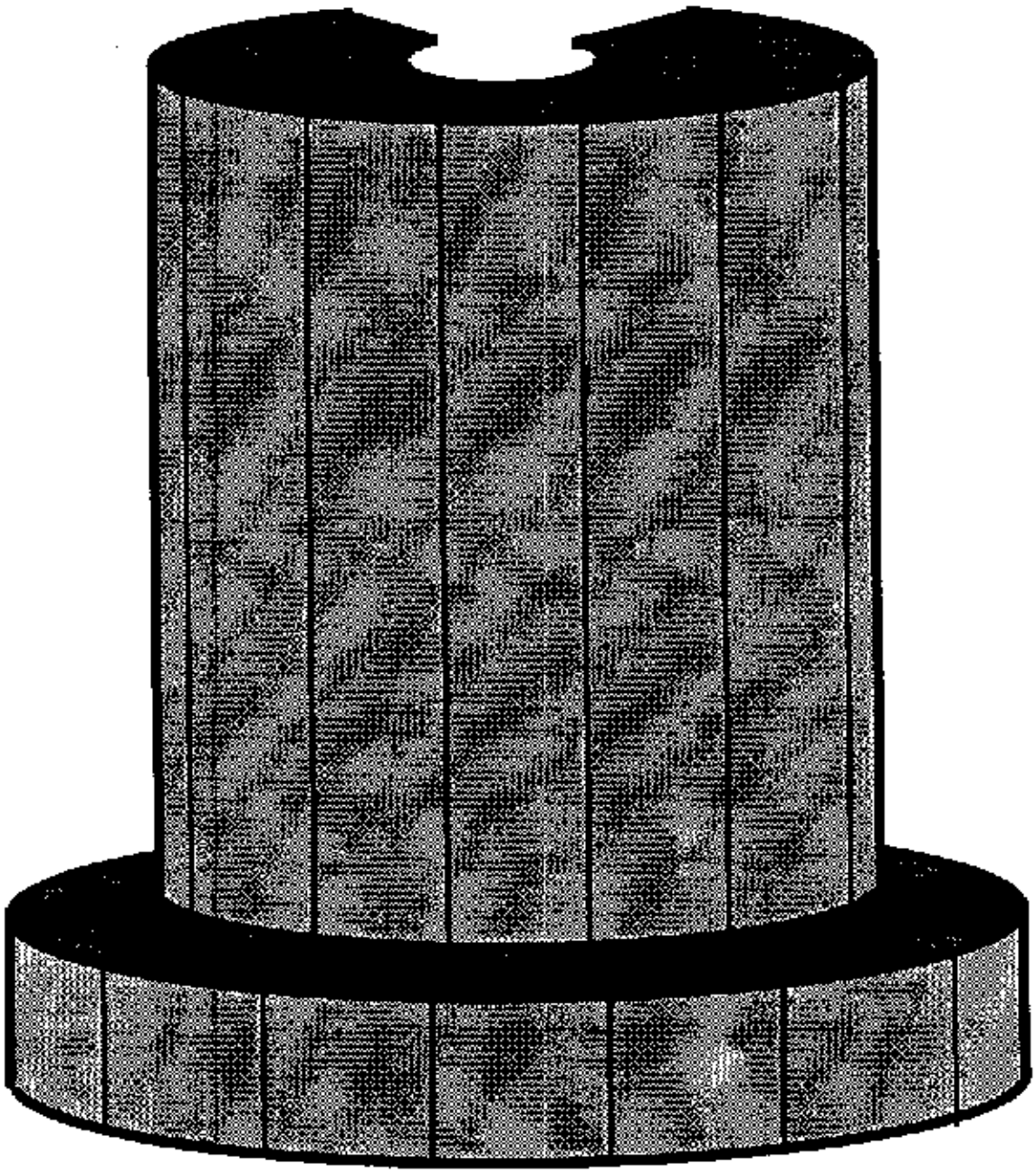
The following different models of the Cray-1 were produced:

CRAY-1A	no IOS or SSD
CRAY-1B	no IOS or SSD
CRAY-1S	optional IOS, SSD available later
CRAY-1M	IOS required, SSD available later

Within each specific model, a number of options existed for memory size, I/O channels, etc.

CRAY-1 SN1 went to Los Alamos National Labs in 1976 and included no software. Serial number two was destined for the National Center for Atmospheric Research (NCAR) but was cannibalized to create a new version of the machine (SN3) which included SECDED memory correction/detection hardware. All CRAY-1s but SN1 contain SECDED hardware which makes them about one foot taller than SN1. Serial number one has been at seven sites during its lifetime and will be retired to the planned supercomputer museum in Chippewa Falls. The last CRAY-1 manufactured was sent to Lockheed in 1983 making 57 the total number of CRAY-1s built.

CRAY-1



CRAY X-MP Supercomputer

The CRAY X-MP line of supercomputers was announced in 1982. The development work was done by Steve Chen and his team of developers. The X-MP system evolved from the CRAY-1 system, taking advantage of newer semiconductor technology and adding improved performance, enhanced architecture, multiple processors, and the SSD. The first system delivered in 1982 was a two-processor system that, at the time, was the upper model in a product line that consisted of both the X-MP and the CRAY-1.

Since then, many enhancements have been made to the X-MP to improve its performance, including adding more memory and more CPUs. All of the current X-MP models are designed with an "extended architecture" (EA) which makes them compatible with the Y-MP computer systems and allows the addition of more memory.

CRAY X-MP Models

The following different models of the X-MP were produced. Each model had a number of options for memory size. Details on options and improvements will be covered later. Only the X-MP EA se models are still produced.

CRAY X-MP/1 one-processor system

CRAY X-MP/2 two-processor system

CRAY X-MP/4 four-processor system

CRAY X-MP EA/1 one-processor system

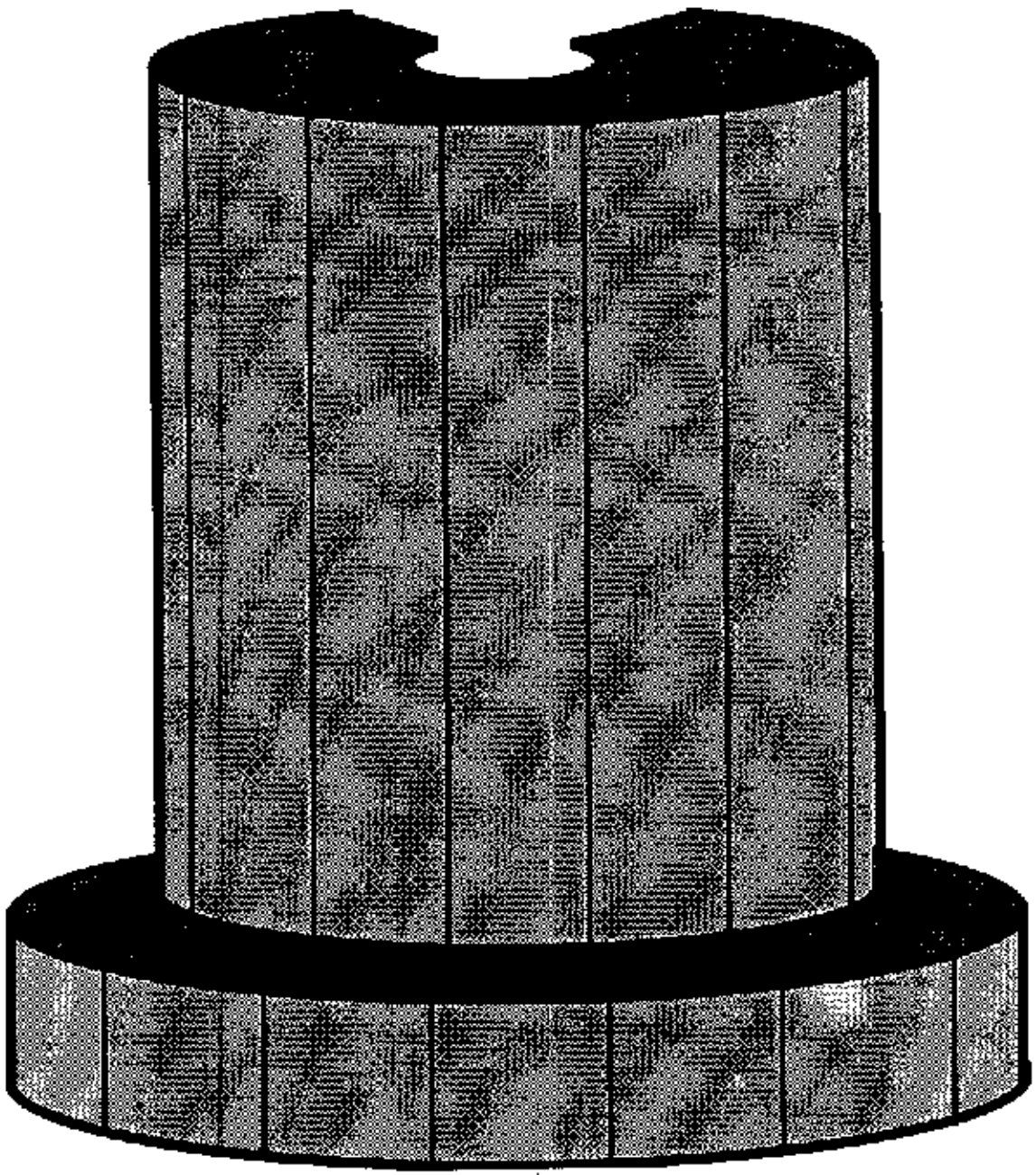
CRAY X-MP EA/2 two-processor system

CRAY X-MP EA/4 four-processor system

X-MP Serial Numbering Scheme

100	X-MP/2	ECL memory
200	X-MP/4	ECL memory
300	X-MP/1	MOS memory (replaced CRAY-1)
400	X-MP/2	MOS
500	X-MP/14se	MOS
1100	X-MP EA/4	MOS
1200	M-MP EA/1 & 2	MOS
1300	X-MP EAse	MOS

CRAY X-MP EA



CRAY-2 Supercomputer

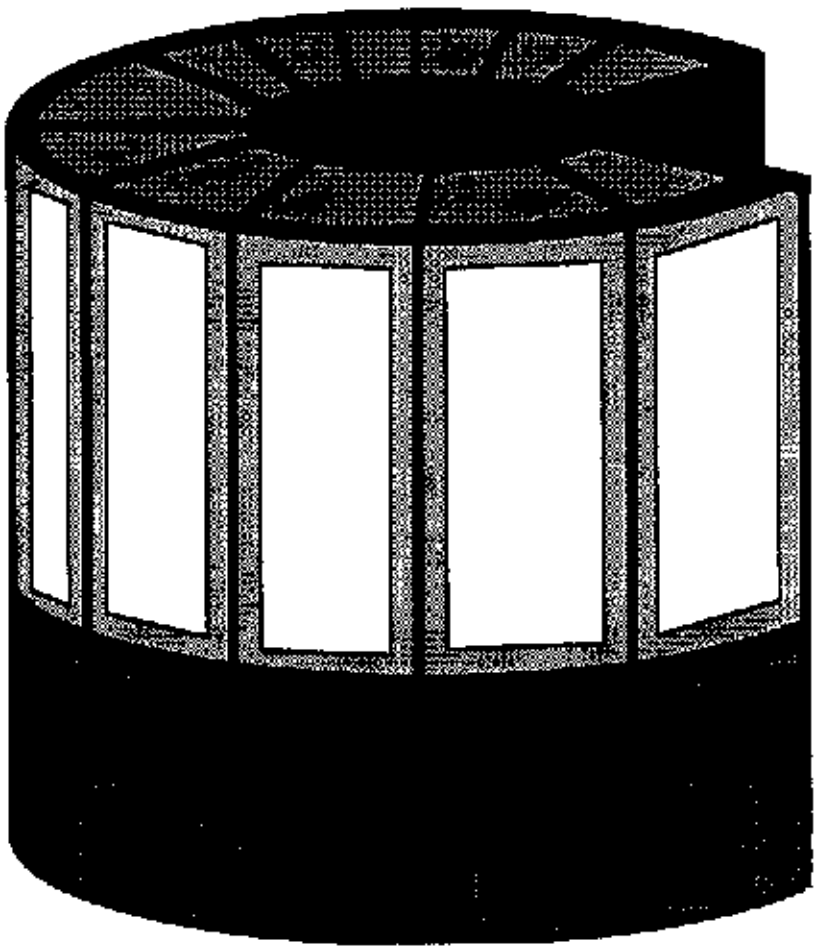
The CRAY-2 system was announced in the summer of 1985. Seymour Cray was responsible for the new design that offered high performance, a very large central memory, and liquid immersion cooling technology.

The CRAY-2 features a common memory of up to 512 million 64-bit words - allowing applications to accomplish in-memory solutions to problems that were previously considered impractical due to much smaller memory sizes.

Four background processors running with a clock period of 4.1 nanoseconds offer an effective throughput six to twelve times that of the original CRAY-1.

The CRAY-2 runs the UNICOS operating system which is based on the AT&T UNIX operating system.

CRAY-2



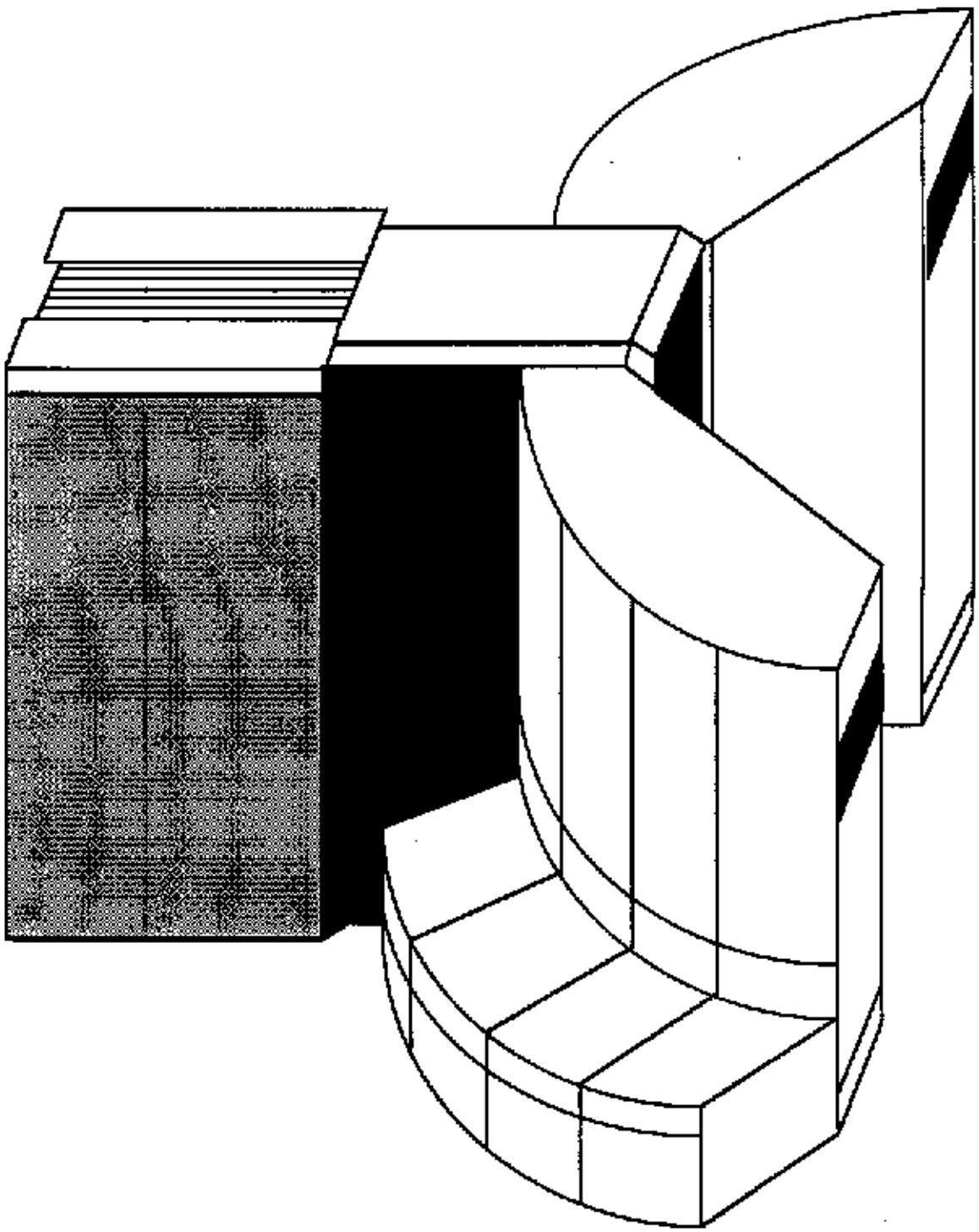
CRAY Y-MP Supercomputer

The CRAY Y-MP line of supercomputers was announced in February, 1988. The Y-MP system evolved from the CRAY X-MP system but it provides more processing power in a smaller space because it incorporates Very Large Scale Integration (VLSI). This essentially means putting more computing power on a chip. Two of the new chips used on the Y-MP contain the processing power of one whole module used in the CRAY-1/A. The first release of the Y-MP has eight CPUs and a clock speed of 6.0 nanoseconds, which is approximately 30% faster than the X-MP's 8.5 nanosecond clock. The Y-MP's design also includes a new cooling design. A new model of the IOS and an SSD both become part of the chassis, combining to form an approximate "Y" shape.

The first Y-MP (SN1001) has 32 million words of main memory and a 128-million word SSD. New models have been announced which will replace most of the X-MP line. Model options will be discussed later.

The intent with the design of the Y-MP was to increase processing power, speed, and memory while remaining compatible with the X-MP line of supercomputers.

CRAY Y-MP



Input/Output Subsystem (IOS)

The I/O Subsystem was introduced in 1979 as an option of the CRAY-1S system. The function of the I/O Subsystem is to reduce the I/O overhead on the mainframe and to provide enhanced I/O capabilities such as on-line tapes. The I/O subsystem is a required component of all CRAY-1M, CRAY X-MP, and CRAY Y-MP systems.

I/O Subsystem Models

The following I/O Subsystem models have been produced:

IOS Model A

IOS Model B

IOS Model C

IOS Model D

Solid-State Storage Device (SSD)

The SSD was introduced in 1982 as an option for the X-MP and CRAY-1S systems. The function of the SSD is to provide an extremely fast secondary storage for the CRAY mainframe. The SSD is viewed by the user and the operating system exactly like a disk. This makes it very easy to take existing I/O-bound programs and speed them up by as much as 20-30 times by moving disk datasets to the SSD.

SSD Models

The SSD comes with a number of memory size options. They are:

32 MWords (can be built into IOS Model C or D)

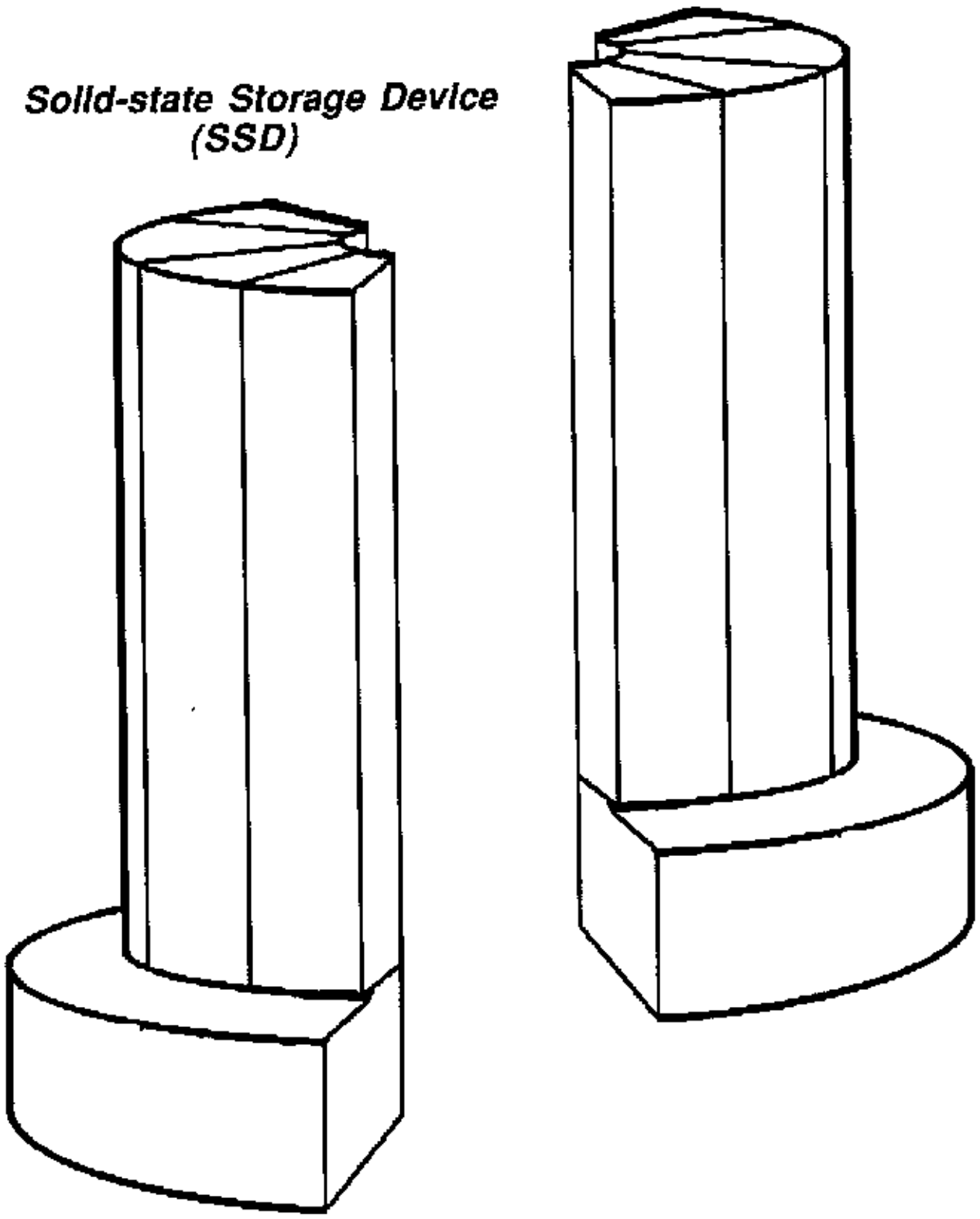
128 MWords (standalone or can be built into IOS Model D)

256 MWords

512 Mwords

Input-Output Subsystem (IOS)

Solid-state Storage Device (SSD)



Other Hardware Products

Front-End and Networking Interfaces

Cray manufactures interfaces for a number of different front-end computers and networks. A front-end interface allows a front-end computer to communicate with a Cray computer. Some of the front ends that are supported are IBM, CDC, Amdahl, and DEC. A network interface provides a means of making a Cray computer a node on a network of many front end computers so they may all communicate with each other.

Disk Storage Devices

Cray does not manufacture disk drives but does supply them with systems. The disk drives in the current product line include the DD-40 (with drives made by Control Data Corp.), the DD-49 (made by Ibis), and the DD-39 (made by Fujitsu). Disk drives that are supported but no longer part of the current product line are the DD-19 and DD-29 drives (made by Control Data).

Tape Drives

Cray provides software and hardware channel support for IBM compatible tape drives as an option on some of the systems. It is normally the responsibility of the customer to obtain the tape drives.

Miscellaneous Peripherals

Every Cray system includes a number of peripheral devices that are used to maintain and operate the system. Depending on the system, these might include operator consoles, small tape and disk drives, and a small printer/plotter.

Power and Cooling Equipment

Because of the tremendous electrical and cooling requirements of a Cray system, some rather large power and cooling devices are supplied with each system. These include motor generators, cooling condensors, and power distribution units. The CRAY-2 with its unique liquid immersion cooling system has additional cooling devices (pods) and coolant reservoirs.

On-site Maintenance Equipment

Each site has the capability to repair systems. Resources include a large spare parts inventory and equipment such as module testers.

Cray Software Products

Operating Systems

The two operating systems that Cray supports are COS (Cray Operating System) and UNICOS (Cray's operating system based on UNIX). Another operating system that runs on many Cray systems is CTSS. CTSS was developed and is supported by a Cray customer.

Programming Languages/Compilers

The programming languages that Cray has developed compilers for are FORTRAN (CFT and CFT77), Pascal, C, and CAL (Cray Assembly Language).

Stations / Networking Protocols

A station is the software that resides on a front-end computer to provide communication with the Cray system. Some of the stations that Cray supports are IBM MVS and VM, CDC NOS and NOS/BE, DEC VAX/VMS, Data General RDOS, and AT&T Unix™. Other front ends are supported with station software available from third parties. Cray now offers the TCP/IP protocol for network communications.

Utilities

A number of utility programs are available for dataset manipulation, editing, debugging, source and object code library maintenance, program loading, etc.

Libraries

A number of libraries exist to provide subprograms in the areas of arithmetic, i/o, scientific routines, utilities, etc.

Applications

Cray Research is not in the primary business of developing new applications programs. However, the applications department assists customers and third party software suppliers in converting and developing programs to run efficiently on Cray systems. The current **Directory of Supercomputer Applications Software** contains hundreds of programs in the areas of structural and mechanical engineering, electrical engineering, chemistry and biotechnology, nuclear engineering, computational fluid dynamics, petroleum reservoir simulation and seismology, mathematics, simulation, graphics and imaging, languages and tools, and others.

THIS PAGE IS INTENTIONALLY LEFT BLANK.

SECTION 2

Cray Product Familiarization

ARCHITECTURAL FEATURES

THIS PAGE IS INTENTIONALLY LEFT BLANK.

General Architectural Features

This section addresses some of the underlying hardware architectural features that contribute to the high performance of Cray systems. Some of the features apply to all Cray systems while others apply to a specific machine.

Some of the concepts covered in this section are:

- **Scalar Processing**
- **Vector Processing**
- **Pipelining of Functional Units**
- **Chaining**
- **Compressed Index/Iota and Gather/Scatter**
- **Multiple Processors**
- **Multiprogramming**
- **Multiprocessing**
  - **at the job level**
  - **at the task level**
    - **macrotasking**
    - **microtasking**

Scalar Processing

Supercomputers, like more conventional computers, are able to do computations in scalar mode. Unlike conventional computers, supercomputers can also compute in vector mode. Before looking at vector processing, it is important to review scalar processing.

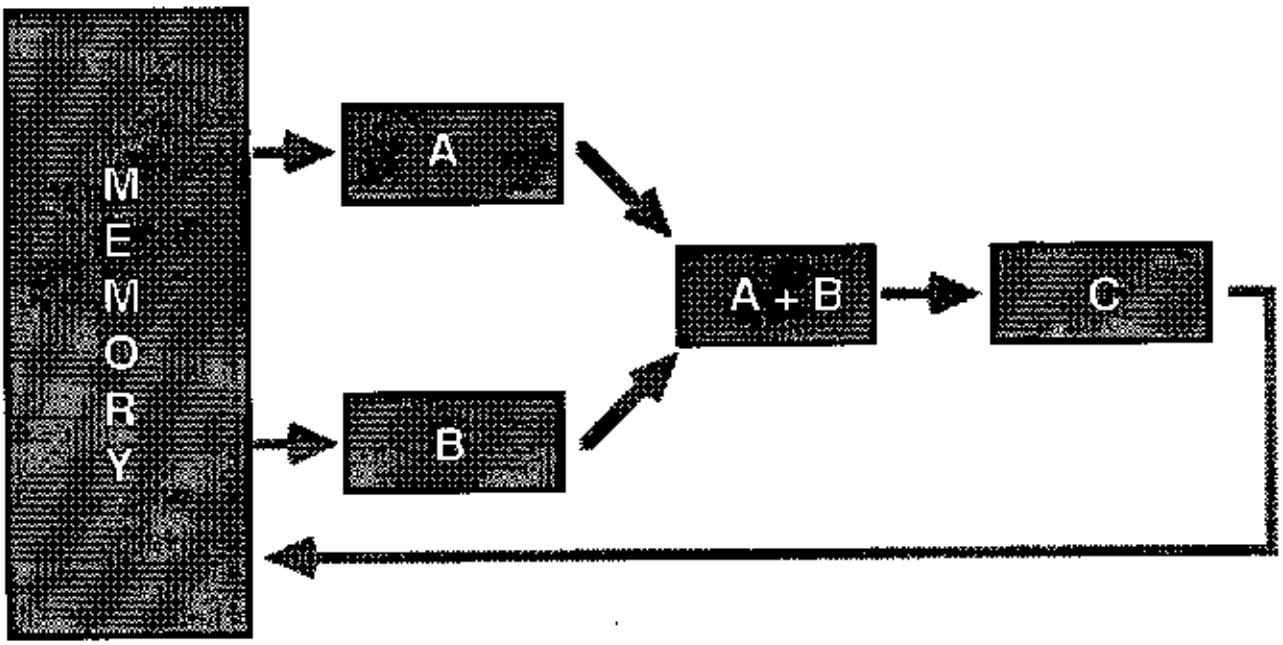
The main concept with scalar processing is that scalar computations deal with a single pair of numbers at one time; that is, one instruction performing one operation at a time. The diagram on the right illustrates a scalar add operation on a Cray computer. Other computers that do scalar processing operate in a similar way.

The diagram on the next page shows the adding of two numbers giving a third number as a result. To do this on the Cray and other computers typically takes four steps (instructions). In this case the original two numbers are in memory and the resulting number will be placed in memory.

Step 1	Load number A into a register
Step 2	Load number B into a register
Step 3	Add A and B putting the result C into a register
Step 4	Store the result C into memory

Although scalar processing does not yield the computational rates that vector processing does, it is important that a supercomputer has high scalar performance. Many scientific problems do not lend themselves to vector processing. Even problems that can be programmed in vector mode still may have large scalar processing requirements. Cray systems provide an excellent balance of high scalar and vector performance.

Scalar Processing



Cray Assembler Language (CAL) example:

```
S1      49      Store the value 49 in S1
S2      51      Store the value 51 in S2
S3      S1+S2   Store the value of S1+S2 in S3
TOTAL,  S3      Store the value S3 at symbolic
                location TOTAL in central memory
                Address TOTAL contains the value 100
```

Vector Processing

High performance vector processing capabilities are what make a supercomputer **super**. The important concept to grasp here is that in vector processing mode, computations are made with groups of numbers (vectors) as opposed to the single sets of numbers processed in scalar mode; that is, one insturction operates several inputs, producing several outputs. The diagram on the right page illustrates a vector add operation on a Cray computer. The actual implementation of vector processing on other systems may be quite different in detail.

The diagram on the next page illustrates adding one group of numbers (vector A) and another group of numbers (vector B) giving a result (vector C). Vector A and B initially reside in memory and the resulting vector will be stored in memory. On the Cray, this typical operation requires four main steps (instructions).

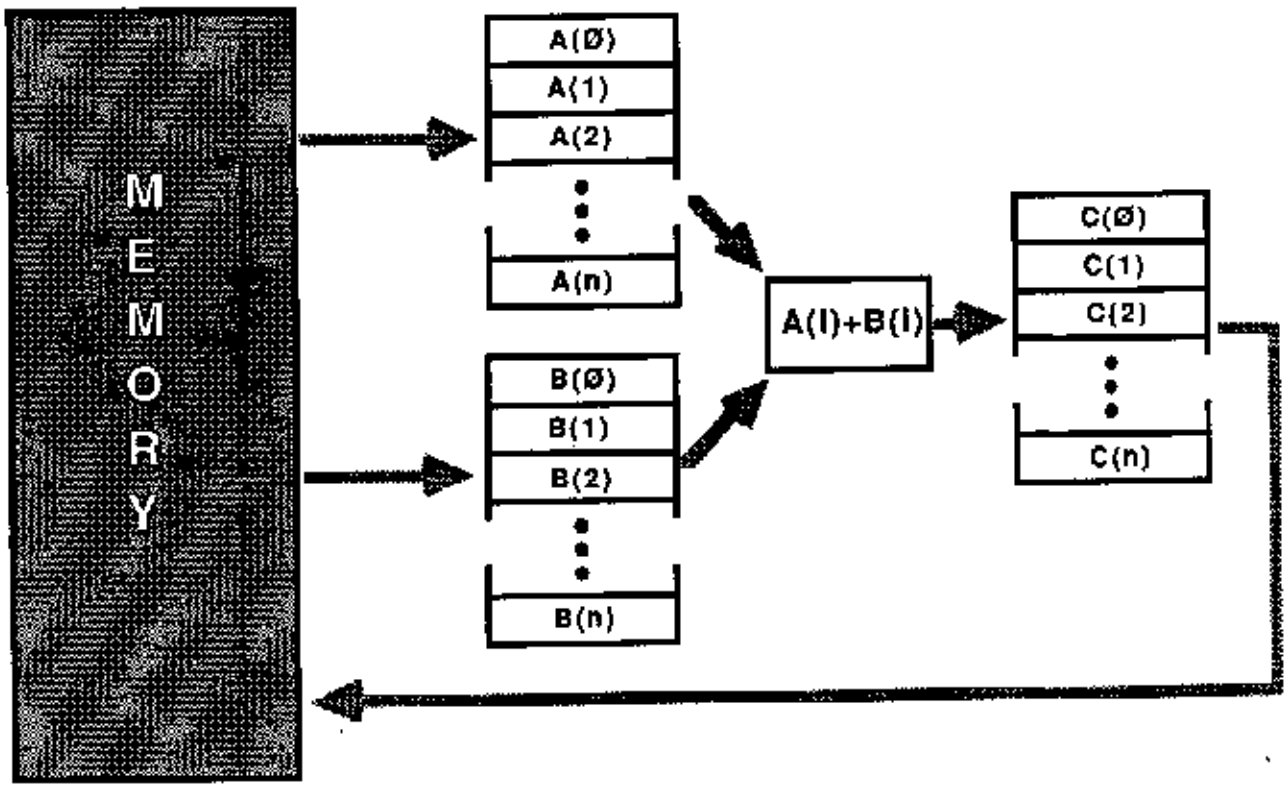
Step 1	Load vector A from memory into a Vector register
Step 2	Load vector B from memory into a Vector register
Step 3	Add the vectors A and B putting the resulting vector C into another vector register
Step 4	Store vector C into memory

In the vector add operation, the first elements of vectors A and B are added giving the first element in vector C. Then, the second elements of A and B are added giving the second element of vector C, and so forth until all the elements of the vectors are processed.

On Cray systems, the number of elements (numbers) that can be processed in a single vector operation is limited by the length of the vector registers. There are eight vector registers in a single Cray CPU, each capable of holding 64 elements. A vector operation may operate on any vector size between 1 and 64.

A feature of the CRAY FORTRAN Compilers (CFT,CFT77) is its abilty to autovectorize programs. The compiler analyzes the program to determine how to most efficiently run it (optimization). The programmer does not need to code for this feature.

Vector Processing



Cray Assembler Language (CAL) example:

(Note that more instructions are required in order to set up the vectors.)

A7	64	Store the value 64 in A7
VL	A7	Make the vector length 64
AØ	VECTOR1	Put the address of VECTOR1 in AØ
V1	,AØ,1	Copy the 64 elements at AØ to V1
AØ	VECTOR2	Put the address of VECTOR2 in AØ
V2	,AØ,1	Copy the 64 elements at AØ to V2
V3	V1+V2	Add V1 to V2 and store result in V3
AØ	VECTOR3	Put the address of VECTOR3 in AØ
,AØ,1	V3	Copy the 64 elements in V3 to the address stored in AØ

Pipelining

The concept of pipelining applies to the design of the functional units in a Cray CPU. A Cray CPU contains a number of functional units, each designed for a specific purpose. For example, there is a separate functional unit to do floating point adds and a separate functional unit to do floating point multiplies. Since the functional units operate independent of each other, operations can occur in parallel (such as floating point adds and multiplies.)

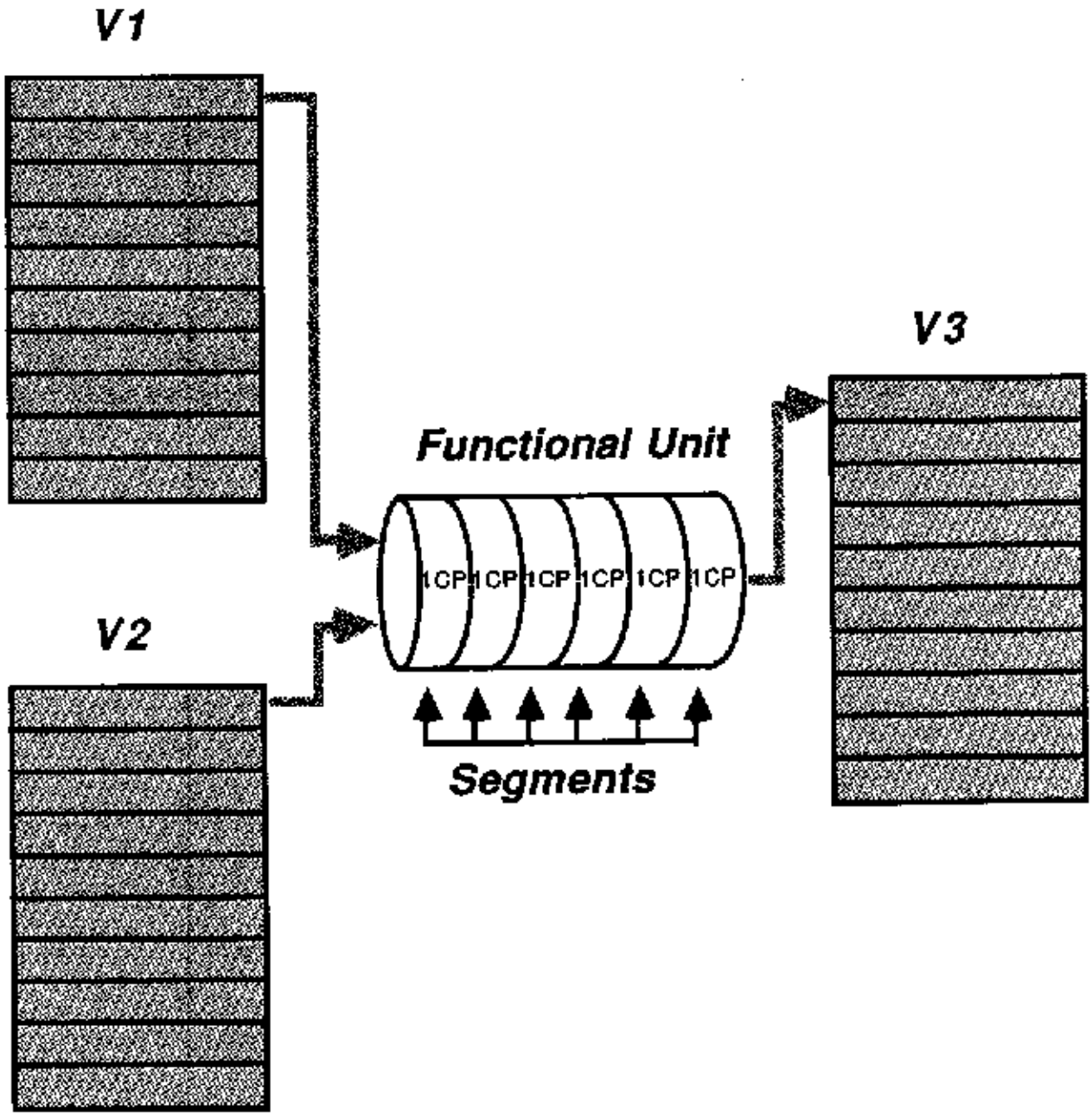
The diagram on the next page illustrates the pipelined design of the functional units. Each functional unit is made up of several separate segments. Each segment is designed to do a part of an operation. The functional unit operates as follows:

In the first clock period of the operation the first elements of vectors V1 and V2 enter the first segment of the functional unit. On the second clock period the elements in the first segment of the functional unit move to the second segment and the second elements of V1 and V2 move into the first segment of the functional unit. This continues each clock period. Soon, results start leaving the functional unit and are entered into vector register V3 at the rate of one result every clock period. This rate can be used in calculating the peak megaflop rate of a particular system (more on this later).

Different functional units will have different numbers of segments depending on their complexity. For example, the reciprocal approximation functional unit has more segments than the floating point multiply functional unit., etc.

Cray terminology for this pipelining design is "fully segmented functional units".

Pipelining



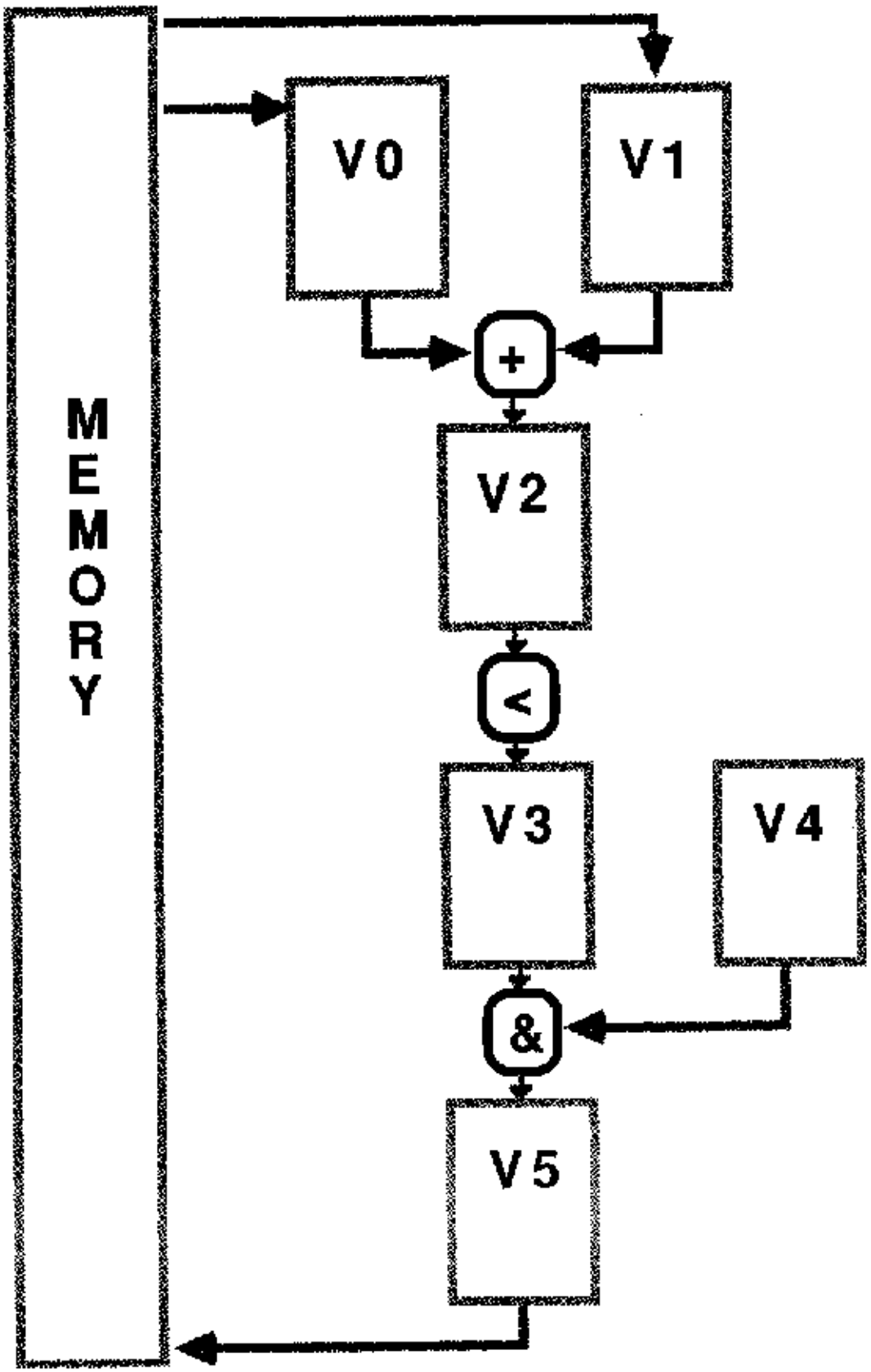
Chaining on the CRAY-1, X-MP, and Y-MP

The functional units in the CPU can function independently (in **parallel**). That is, an add could be taking place at the same time as a multiply. Because of this possibility, output from one operation can become input to another functional unit.

For example, the diagram on the next page shows two vector registers being loaded. Based on the pipelining function, as soon as VØ and V1 each contain one element, those elements may begin the process of being added together. Likewise, as soon as one element has been put into V2 it can go through the shift process and be put into V3 (a **shift** does things like chop off leading zeros, etc.).

Assuming V4 was already loaded, V3 and V4 can be **anded** (anding does a logical compare between the elements in V3 and V4 and comes up with a new number). The results can begin being written to memory as soon as one element has been put into V5.

Chaining on a Y-MP



COMPRESSED INDEX/IOTA AND GATHER/SCATTER

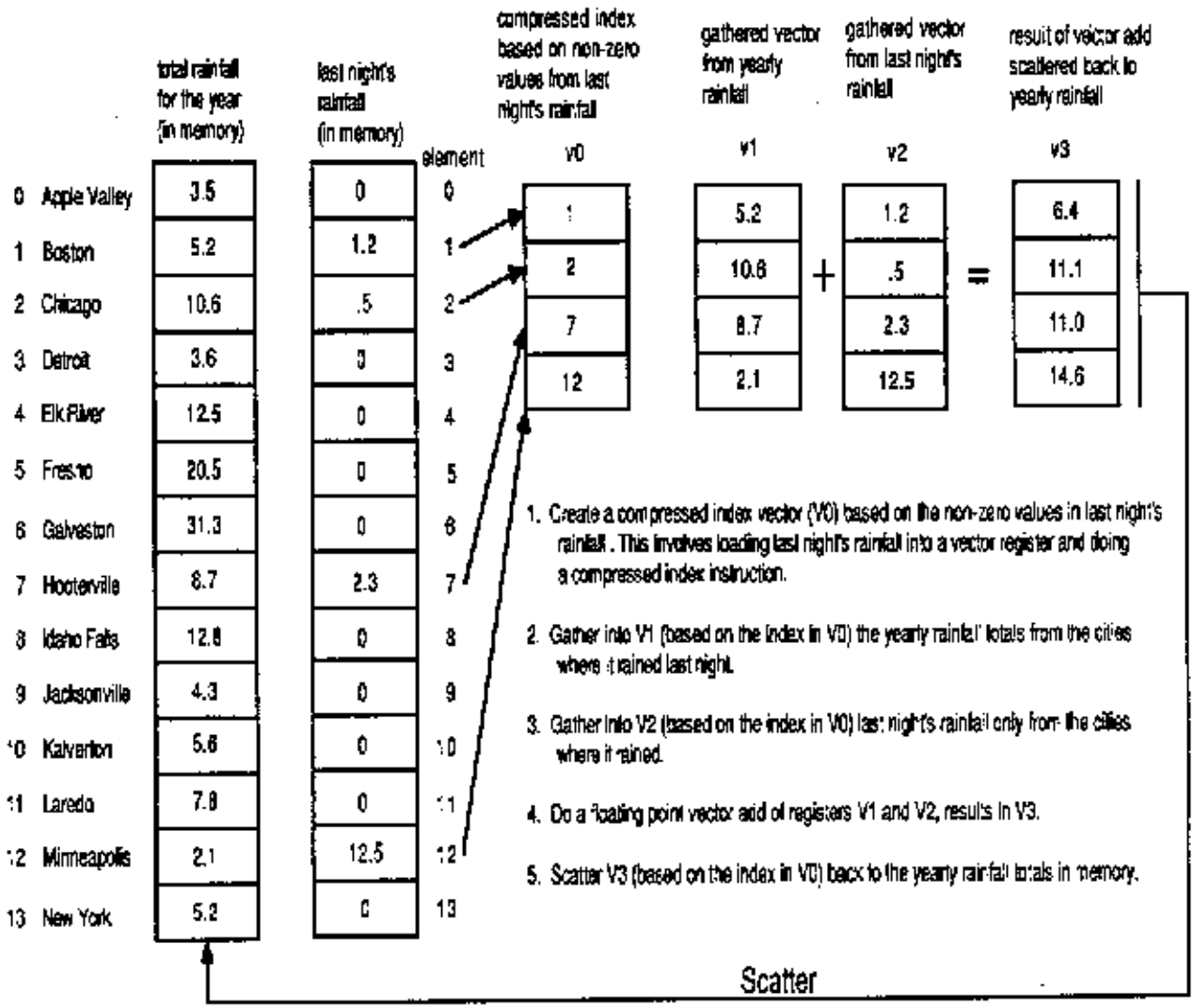
Some supercomputer programs such as those used in weather forecasting and in predicting where oil may be located deal with very large collections of data. Those collections often contain a lot of unuseable or unnecessary data. In order to more efficiently run the programs utilizing only the important data, Cray computers now offer an efficient means of locating it in memory.

Compressed Index is a means of creating an index of the location of the important data. An example might be to make an index of all the locations which contain a non-zero value because the zeros are useless data for our program. By avoiding the locations containing zero we avoid wasting computer time dealing with them. A term used for groups of data which contain lots of unnecessary data is **sparse matrix**. Compressed index is know as **compressed iota** on the CRAY-2.

Once the index is created, a **gather** operation may be done which simply uses the index to find the non-zero data and copy it. Once copied, the data can be run through a functional unit to create new values. The new data (results) can then be copied back into the exact locations it originally came from by using the same index. This is called a **scatter** operation. Now the new data can be gathered using the same index and run through other functional units to come up with more new values, if needed.

Another term associated with gather/scatter and compressed index is **stride**. It simply is a way of setting up an index that allows gathering or scattering data on a fixed increment (e.g., every fifth data element). This is useful when the data dealt with is known to contain important data only specific incremental locations.

Gather / Scatter / Compressed Index Example



Cray Assembler Language (CAL) example

```
AØ      NITERAIN    AØ gets address of NITERAIN array
V1      ,AØ,1       V1 gets a vector length of NITERAIN, stride=1
VØ,VM   V1,N        VØ gets index of non-zeros in V1
S1      VM          S1 gets vector mask
A1      PS1         A1 gets population count of S1
VL      A1          Vector length set to A1 (= # of non-zeros)
V1      ,AØ,VØ      V1 gathers non-zeros from NITERAIN
AØ      YEARRAIN    AØ gets address of YEARRAIN
V2      ,AØ,VØ      V2 gathers non-zeros from YEARRAIN
V3      V1+FV2      V3 gets floating point sum of V1 and V2
,AØ,VØ  V3          V3 scattered back to YEARRAIN
```

Multiprogramming

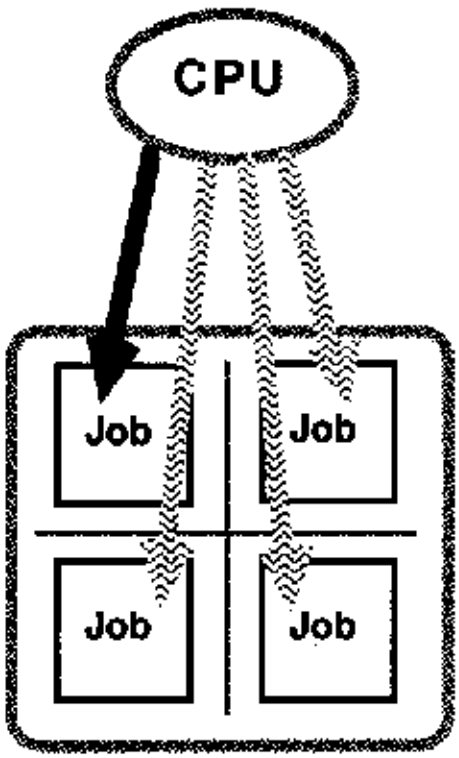
A computer would be very slow if it ran one program to completion before starting the next. Multiprogramming means that many programs can run **concurrently** on the system - each program is given a "time-slice" and then waits while another runs. This is also known as **interleaving**.

Throughput is improved by multiprogramming even though there is more system overhead involved. This is because of **parallel** system functions (several parts of the computer can be running at the same time).

In the case of multiprogramming, the output of one program can be routed back to a front-end computer while an incoming program is being loaded into memory and still another is executing ("number-crunching") in the CPU. The operating system keeps track of each job. It will assign a hardware component to complete a task and then, instead of waiting for that task to finish, assign another idle component a task.

This requires a sophisticated operating system, but does not require the applications programmer to call for it.

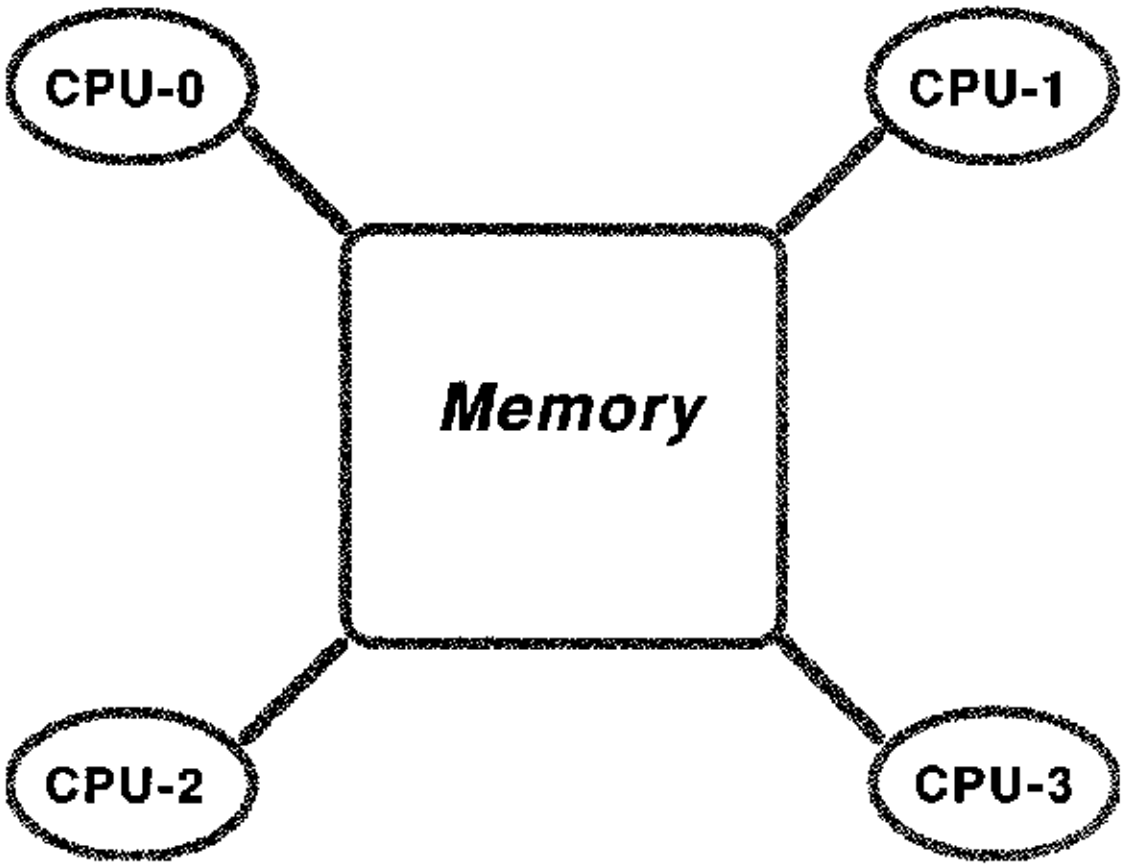
Multiprogramming



Multiprocessing

If one CPU is good, two or more should be better and that is true with the X-MP, CRAY-2, and Y-MP. Each CPU has its own identical set of functional units and registers operating in parallel. Multiprocessing takes that concept one step further by incorporating two or four CPUs in one machine. This increases the capacity of the machine as well as program throughput.

Multiple Processors

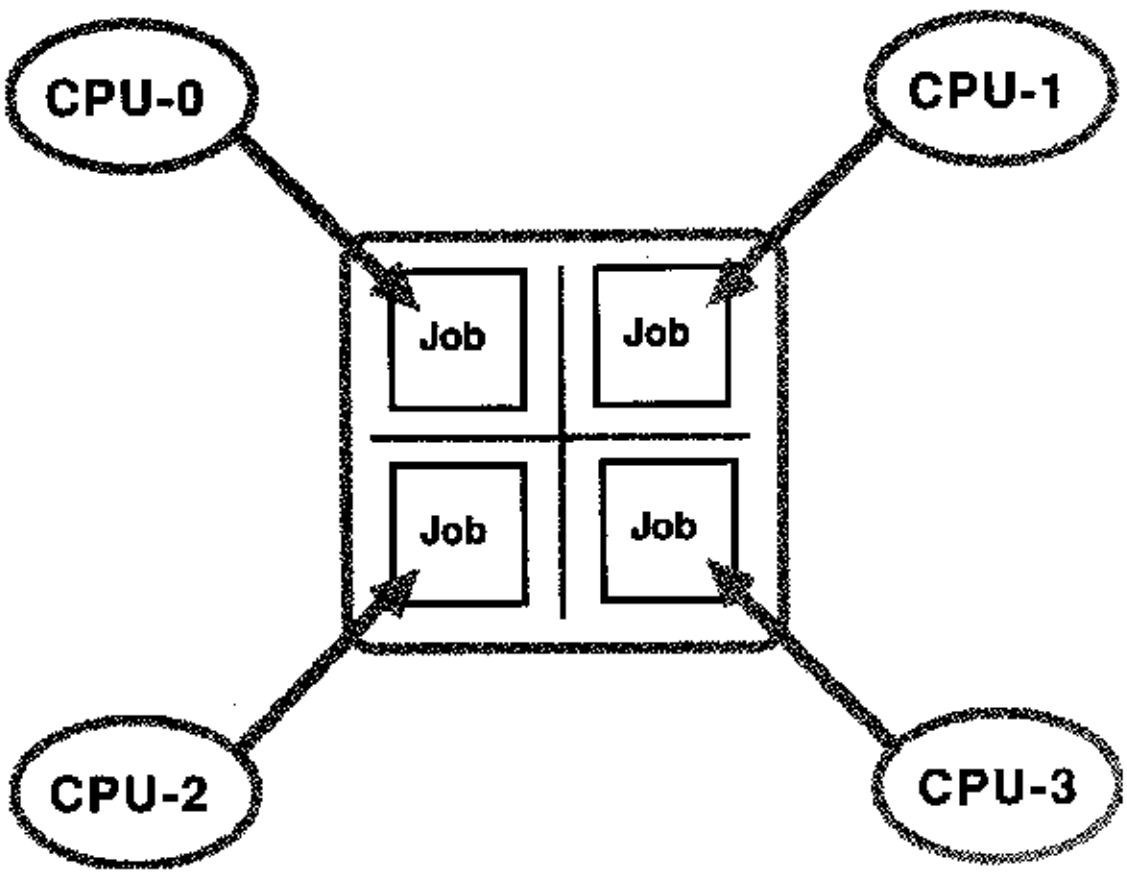


Multiprocessing at the job level

A machine with more than one processor can divide up the work load. With multiprocessing, incoming programs can be assigned a CPU. This allows for up to eight programs running in separate CPUs at any one time (on a Y-MP).

Multiprocessing requires the operating system to ensure that a program in one CPU does not conflict with a program in another CPU.

Multiprocessing at the job level



Multiprocessing at the task level (Macrotasking)

Imagine a program so large that it alone could use a whole system's resources for hours, days, or even weeks. Because there are programs like that, it would be more efficient to divide up the workload among more than one CPU.

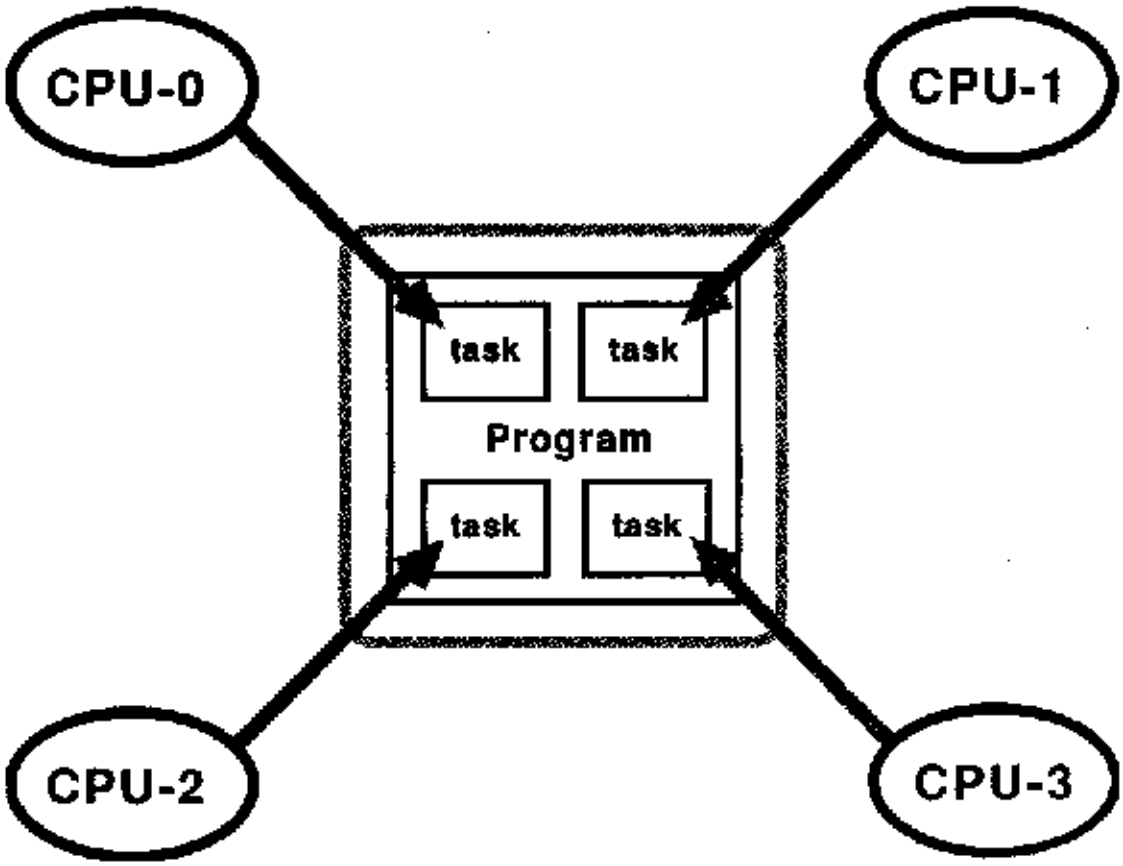
The means for doing that is known as **multitasking**. This feature allows two or more parts of a program (tasks) to be executed in parallel which results in faster program execution than if the program ran serially.

Experience shows that multitasked applications running on CRAY X-MP/2 computers can realize speed increases of 1.8 to 1.9 times over single-processor CRAY X-MP execution times; speed increases of 3.5 to 3.8 times have been achieved with CRAY X-MP/4 systems. The throughput of a multitasked application which utilizes all of a Y-MP's CPUs could be more than seven times faster than if it were run in only one of the Y-MP's CPUs.

One method of multitasking on a CRAY X-MP, CRAY-2, or Y-MP is called **macrotasking.**

To use macrotasking the program is modified to use FORTRAN-callable subroutines which define and synchronize tasks within subroutines. The system launches subjobs which compete for CPUs. For this reason macrotasking is best suited to programs with large tasks running with dedicated processors.

Multiprocessing at the task level (macrotasking)



Multiprocessing at the task level (Microtasking and Autotasking)

Two other forms of multitasking available for use on CRAY computers are microtasking and autotasking. Microtasking is similar to macrotasking in that it divides up a single application program's work among two or more processors.

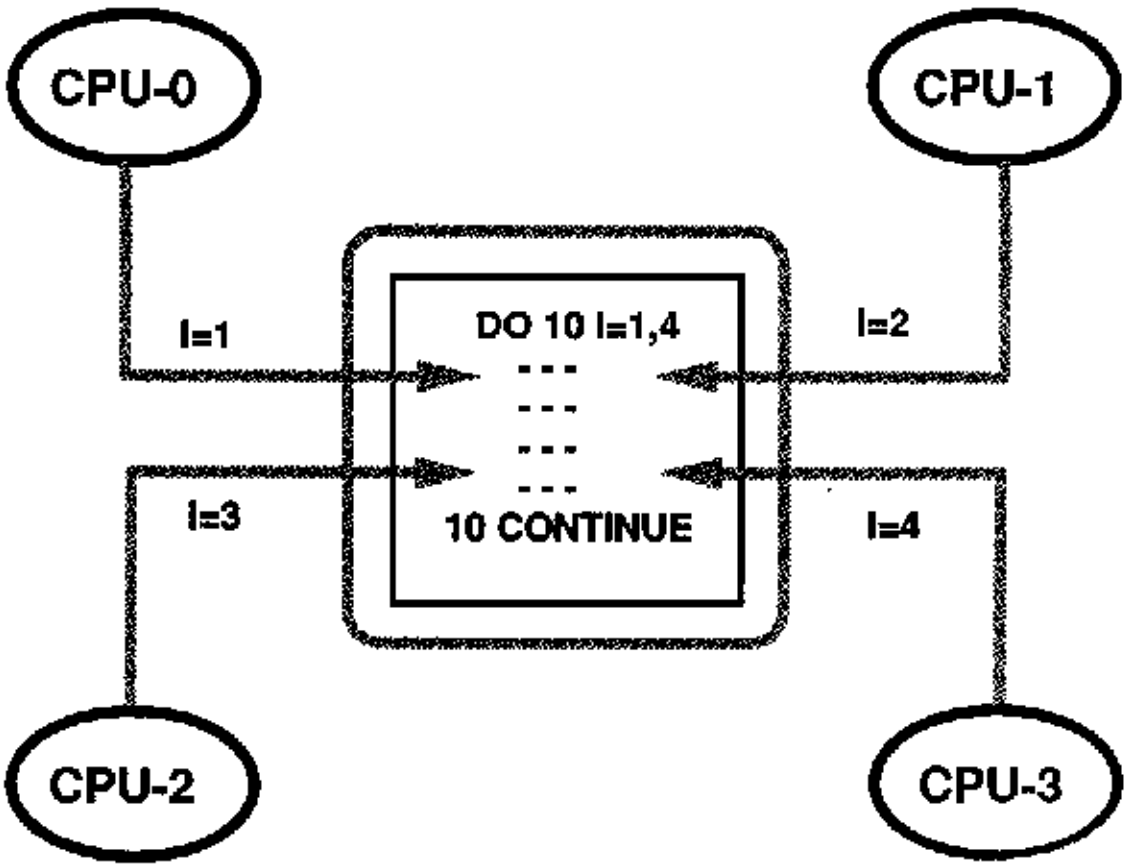
It is different, however, in that it is a dynamic function which operates at run-time. Instead of specifying portions of the program's code to be handled by different CPUs (as macrotasking does), microtasking looks for idle CPUs which it could use for parts of the program. If it can't find any, it will execute in its current CPU. This helps efficiently use the system's resources and allows faster execution of a program even on a mainframe in a production environment (many users).

Microtasking is primarily used with FORTRAN DO Loop-level routines. A DO loop is a series of computations done over and over a specified number of times. Microtasking uses a FORTRAN preprocessor called PREMULT that identifies DO loops which could be run independently (in parallel) on separate processors. As the program runs, each of the identified DO loops is executed in the next available CPU. PREMULT has its own efficiency monitor to allow observation of speed improvements. Microtasking uses compiler-like directives which makes it easy to use.

Microtasking is beneficial for programs with any size task running in either a dedicated or production (many users) environment.

Autotasking does essentially all of the above steps in multitasking a program, but it does it automatically, with no user intervention required. It requires several passes through a program to analyze it, but it divides up the work as it sees fit. Autotasking can also include some microtasking code for special situations.

Multiprocessing at the task level (microtasking and autotasking)



CLOCK RATE and MEGAFLOPS

Clock Rate and Megaflops (MFLOPs) are two terms often used in rating a supercomputer's processing speed. **Clock rate** (or clock period) refers to the timing or pulse of the computer. The smaller the number, the faster the rate at which the hardware can theoretically work. Of course, the distance between hardware components is critical for timing. That is why Cray computers are so compact, using very short wires and a circular design for instance. The number of ticks per second can be determined by the formula 1/clock speed (e.g., on a Y-MP with a 6.0 nanosecond clock the formula is $1/6.0^{-9}$ which equals 166 million ticks per second).

Supercomputer speed is often rated in **megaflops** rather than ticks per second. Megaflop stands for "millions of floating point operations per second". Floating point numbers include numbers which contain a decimal point (e.g., 132.098) and large numbers which are more easily defined using exponential notation (e.g., 10^{23}). Since these types of numbers are more often used in supercomputer applications than simple integers (whole numbers), how fast they can be added, multiplied, etc. is considered a good means of measuring the speed of the computer.

One floating point operation per second would be like completing this operation (1.23+8.1=9.33) in one second. One megaflop would be one million of them per second. Because Cray computers incorporate multiprocessing and chaining, the following formula is used to determine megaflops on Cray computers: Ticks per second x number of CPUs x number of functional units running in parallel.

CLOCK RATES AND MFLOPS

	Clock Rate (ns)	Ticks/ Second (millions)	MFLOPS (1 F.U.)	MFLOPS (2 F.U.)	MFLOPS (2 F.U.) x 2 CPUs	MFLOPS (2 F.U.) x 4 CPUs	MFLOPS (2 F.U.) x 8 CPUs
CRAY-1	12.5	80	80	160	- -	- -	- -
CRAY X-MP EA	9.5	105	105	210	420	840	- -
	8.5	117	117	234	468	936	- -
CRAY-2	4.1	243	243	486	972	1944	- -
Y-MP	6.0	166	166	332	664	1328	2656

CRAY LOGIC AND MEMORY CHIP TECHNOLOGY

Probably the two computer components that have had the greatest impact on the speed and size of Cray computer systems are the logic and memory chips.

LOGIC CHIPS

Logic chips contain the circuitry (on/off switches) which, when used in certain combinations, perform all the possible instructions a Cray computer is designed to do such as calculating answers to mathematical equations. The chips are specifically designed to perform these functions by using a branch of mathematics known as Boolean Algebra (which was invented by George Boole in the 19th century who believed that everything in the universe could be explained logically in terms of man and God (or 1 and Ø)).

CRAY-1 systems used simple 5/4 AND gates in combinations. These chips contain only two gates (switches). Because of this, it took 550 2-board modules to make one CPU. The X-MP and all CRAY-2 systems were designed to use 16-gate logic chips. This, in effect allows eight times the logic on one chip, although not all 16-gate chips use all 16 gates. This chip density effectively decreased the size of a CPU in half.

Currently, the Y-MP and the X-MP EA incorporate 2500-gate array logic chips. These chips are roughly equivalent to 1250 5/4 AND gate chips and one chip is roughly equal to one X-MP module. Because of these chips, one CPU now fits on one Y-MP module, although the module is larger than an X-MP module. The Y-MP uses these chips exclusively and the X-MP EA uses some 2500-gate and some 16-gate chips. Some of the 2500-gate chips are produced in Chippewa Falls. The 2500-gate array chips are considered VLSI (Very Large Scale Integration) chips.

MEMORY CHIPS

As with logic chips, memory chip technology has improved over time. Today's memory chips are much denser (store more data) and faster (to read and write the data) than the memory on the CRAY-1. The CRAY-1A used 1 Kbit memory chips; today an SSD uses 1 Mbit chips. One Y-MP memory module equals 72 1Mbit SSD memory chips which equals **8 columns** on a CRAY-1A.

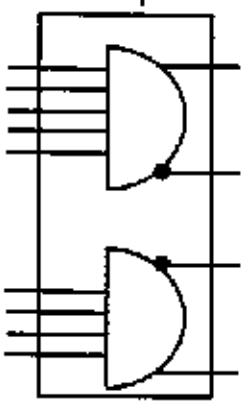
Cray computer systems have used different types of memory. One type is called **ECL (Emitter-coupled logic) memory**. These chips are quite fast, but they require a lot of electrical power, generate a lot of heat, and take up a lot of space. Capacity of ECL memory chips used on Cray computers ranges from 1Kbits to 256Kbits per chip. ECL chips on the Y-MP have an access time of 15 ns.

Another type is called **MOS (Metal-oxide semiconductor) memory**. MOS memory is cheaper, denser, requires less power, and generates less heat than ECL memory. But, it is also slower. Capacity of MOS chips used on Cray computers ranges from 16Kbits to 1Mbits per chip. There are two types of MOS chips used on Cray computers: DRAM and SRAM. **DRAM (Dynamic Random Access Memory)** chips are cheap, but use capacitors to hold the electrical charge representing the 1 or Ø. This requires overhead for the computer to recharge the capacitors to hold their value. DRAM chips on the CRAY-2 have an access time of 80 ns. **SRAM (Static Random Access Memory)** chips do not need the refresh and so are faster. SRAM chips on the CRAY-2 have an access time of 55 ns.

CRAY CHIP TECHNOLOGY HISTORY

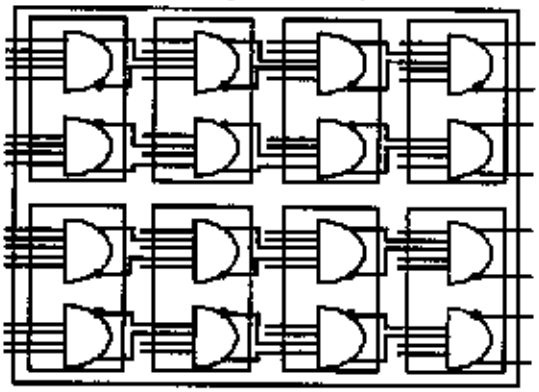
LOGIC

5/4 AND gate chip



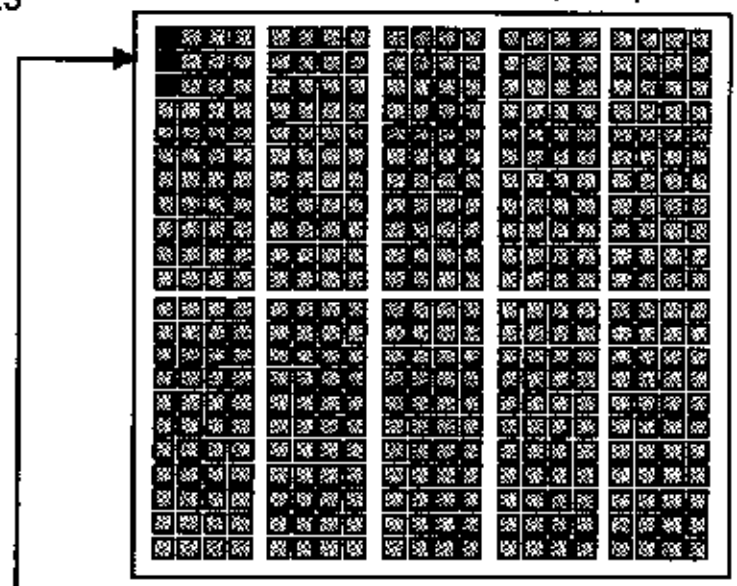
DØ=CØ E5 E4 E3 E2
D1=C1 E5 E4 E3

16-gate chip



DØ=C5 C4 C3 C2 C1 CØ E5 E4 E3 E2 E1

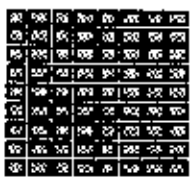
2500 macrocell array chip



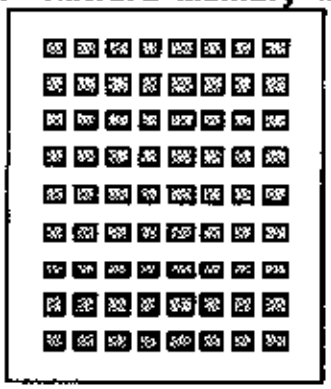
DØ=C5 C4 C3 C2 C1 CØ E5 E4 E3 E2 E1

MEMORY

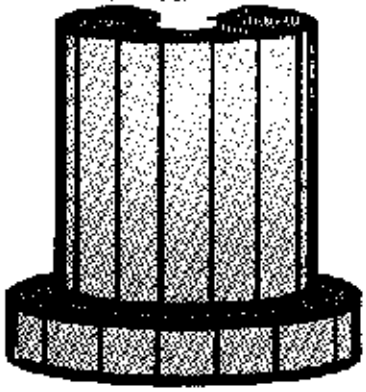
72 1Mbit chips



1 Y-MP 1Mword Memory Module



8 Columns of a CRAY-1A



CRAY PRODUCT STRATEGY

- Deliver the fastest vector and scalar processors
- Parallel processing via multitasking and microtrasking
- Commitment to large real memories
- Support of very high-speed I/O
- Fast, efficient software
- System integration supporting a variety of mainframes
- Expanding application software
- On-site support and maintenance
- Close customer contact
- Large R & D expenditures for hardware and software

Cray System Comparisons

Model		1/A,B	1/S	1/M	X-MP 14se	X-MP 1	X-MP 2	X-MP 4	CRAY-2	CRAY-2S	X-MP EA/1se	X-MP EA/1	X-MP EA/2	X-MP EA/4	Y-MP2	Y-MP4	Y-MP8
CPU	Number of CPUs	1	1	1	1	1	2	4	4	2,4	1	1	2	4	1,2	2,4	4,8
	Clock period (ns)	12.5	12.5	12.5	10.0	9.5, 8.5	9.5, 8.5	9.5, 8.5	4.1	4.1	8.5	8.5	8.5	8.5	6.0	6.0	6.0
Memory	Size (mw)	.5	.5,1, 2,4	.5,1, 2,4	4	4, 8, 16	2,4 8,16	4, 8, 16	256, 512	64, 128	4, 16	16,32, 64	16,32, 64	16,32, 64	16,32	16,32, 64	32,64, 128
	Memory chip technology	ECL	ECL	MOS	MOS	MOS	ECL, MOS	ECL, MOS	Dynamic MOS	Static MOS	MOS	MOS	MOS	MOS	ECL	ECL	ECL
	Banks	8,16	8,16	8,16	16	16,32	16,32	32,64	128	64, 128	16	32, 64	32, 64	32, 64	64	128	256
	Memory ports	1	1	1	5	5	2x4	4x4	4x1	2x1, 4x1	5	5	2x4	4x4	1x5, 2x4	2x4, 4x4	4x4, 8x4
I/O	I/O subsystem	No	Opt.	Opt.	Built-in	Yes	Yes	Yes	FGP	FGP	Yes	Yes	Yes	Yes	Yes	Yes	Yes; Opt. 2nd
	SSD channel (Mbytes/s)	-	100	100	NA	100, 1000	1000	2x 1000	No	No	NA	1000	1000	2x 1000	1x 1000	2x 1000	2x 1000
	High-speed channel pairs (100 Mbytes/s)	-	1,2	1,2	1,2	1,2	2	4	-	-	1,2	1,2	2	4	1,2	1,2,4	1,2,4
	Low-speed channel pairs (6 Mbytes/s)	12	12	12	4	4	4	4	-	-	4	4	4	4	1,2	1,2,4	1,2,4
	Mainframe Columns	12,8	12,8	12,8	6	6	12,8	12	14	14	6	6	12,8	12	1	1	1

THIS PAGE IS INTENTIONALLY LEFT BLANK.

SECTION 3

Cray Product Familiarization

CRAY Y-MP

Computer System

Cray Research, Inc. Software Training

THIS PAGE IS INTENTIONALLY LEFT BLANK.

CRAY Y-MP SUPERCOMPUTING PHILOSOPHY

- Innovative VLSI architecture is incorporated in:

  CPU technology
  - same functionality as X-MP using less real estate

  Hardware design
  - yields significant performance improvements

  Memory accessing
  - larger numbers of paths to central memory

- Improved reliability

  Fewer parts

- I/O capability

  Second I/O subsystem available option allows

  Unprecedented amount of accessible disk space

- Wide variety of memory and SSD options
- Compatible with CRAY X-MP

  CRAY X-MP (X-mode) instruction set may run on the CRAY Y-MP

CRAY Y-MP SERIES CHARACTERISTICS

- Highly modular architecture in three frame types
- 1, 2, 4, or 8 CPUs
- 6 nanosecond clock period
- 16/32/64/128 Mwords Central Memory
- Extensive use of VLSI logic circuits

Y-MP Central Memory

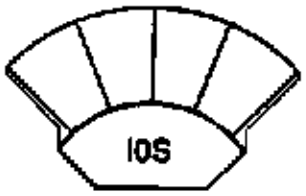
- Up to 128 million words, depending on model, enabling users to solve large problems without the need for as much disk I/O
- Directly addressable for faster reads and writes
- Memory size is upgradable on most models
- Memory integrity is checked using single-bit error correction, double-bit error detection (SECDED) logic
- Multiprocessor Y-MPs share a central memory organized in interleaved memory banks that can be accessed independently and in parallel during each machine clock period
- Each processor has four parallel memory ports connected to central memory: two for vector and scalar fetches, one for result store, and one for independent I/O operations (such as to the IOS and SSD); the CRAY-1system had only 1 bidirectional memory port
- The multiport memory has built-in conflict resolution hardware to minimize delays and maintain the integrity of simultaneous memory bank references
- 15ns 64Kbit or 256Kbit ECL memory is or will be used on all Y-MP models

CRAY Y-MP SERIES CONFIGURATIONS

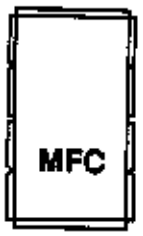
Product Family	Frame	CPUs	1989/1990 Memory Options* 128 Mword	64 Mword	32 Mword	16 Mword
Y-MP	8	8	*Y-MP8/8128*	Y-MP8/864	Y-MP8/832	
		4	*Y-MP8/4128*	Y-MP8/464	Y-MP8/432	
Y-MP	4	4		*Y-MP4/464*	Y-MP4/432	Y-MP4/416
		2		*Y-MP4/264*	Y-MP4/232	Y-MP4/216
		1		*Y-MP4/164*	Y-MP4/132	Y-MP4/116
Y-MP	2	2			*Y-MP2/232*	Y-MP2/216
		1			*Y-MP2/132*	Y-MP2/116

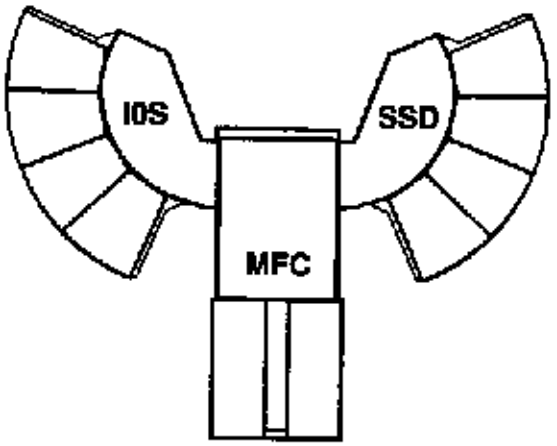
* Models in italics will be available for delivery in the second quarter of 1990.

CRAY Y-MP FOOTPRINTS

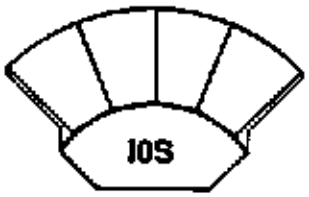


CRAY Y-MP2

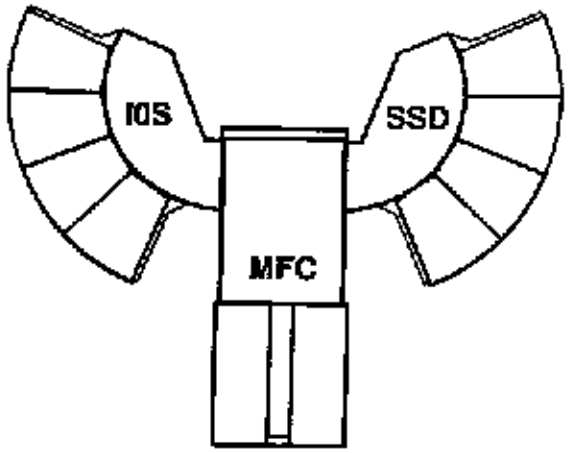




Y-MP4 or Y-MP8



CRAY Y-MP8 with 2nd optional IOS



VLSI ARCHITECTURE

The VLSI architecture used on the Y-MP is based on 2500-gate macrocell array chips. Because of the use of these chips the following is possible in the Y-MP:

- Reduced module count
- Improved reliability
- Lower usage of costly floor space
- Faster repair capability
- A faster clock period

Y-MP MODULES

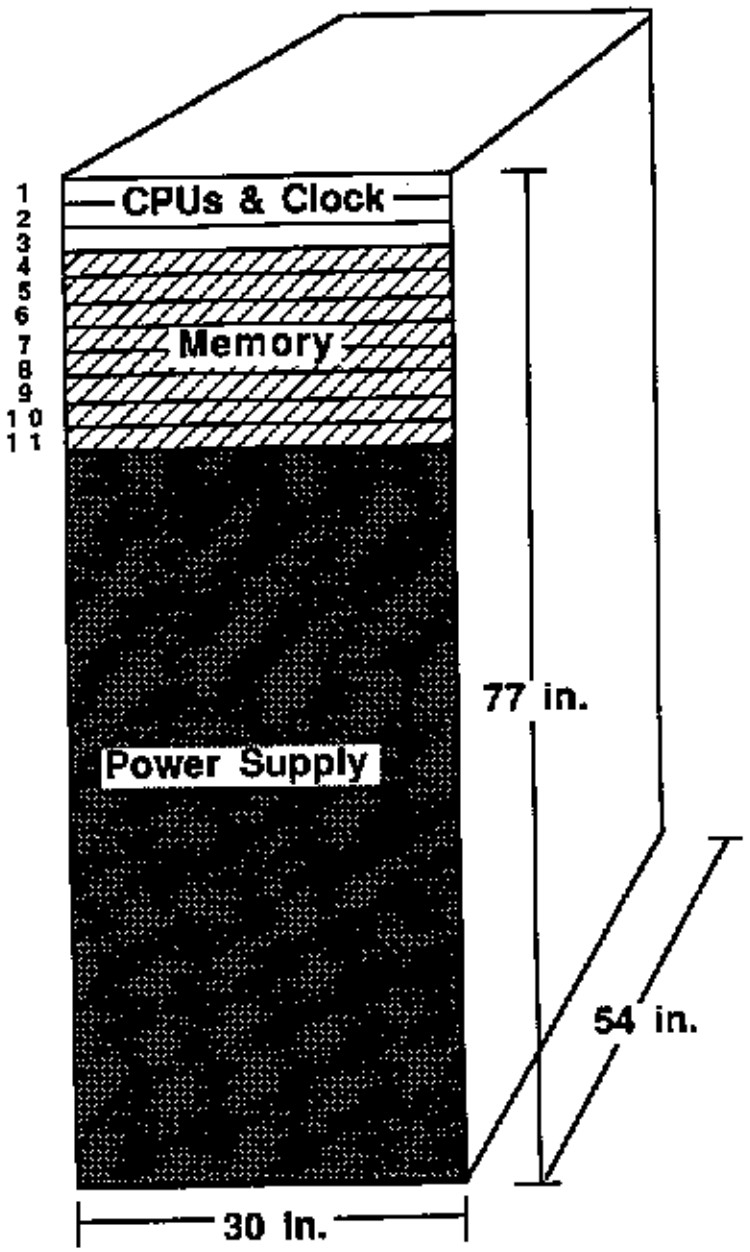
- Each module:
  - measures 23.3 in. x 12.8 in. x 1.4 in.
  - consists of four printed circuit boards and cold plate assemblies
    - cold plate assemblies are made up of two hollow cold plates with a printed circuit board mounted on each side of the cold plates
    - printed circuit boards have 12 layers and measure 11 in. x 21.2 in. Each board contains 4 million traces and spaces, and 8,000 buried resistors

Y-MP2 CHASSIS COLUMN

- Total of 11 slots in one column
  - 1 or 2 CPU modules
  - Up to 8 memory modules
  - 1 system clock module

Y-MP2 CHASSIS DIMENSIONS

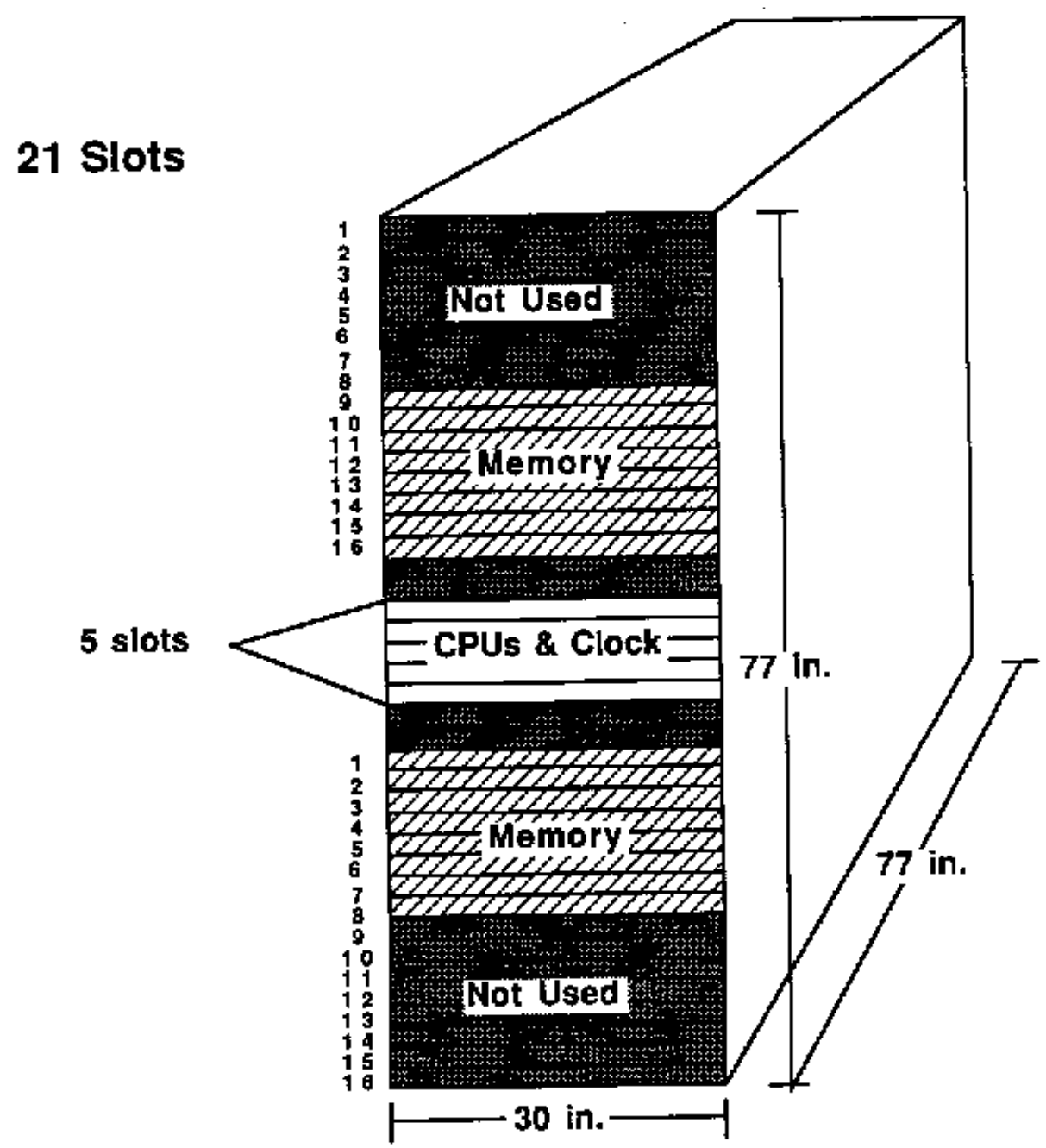
11 Slots



Y-MP4 CHASSIS COLUMN

- Total of 21slots in one column
  - 1, 2, or 4 CPU modules
  - Up to 16 memory modules
  - 1 system clock module

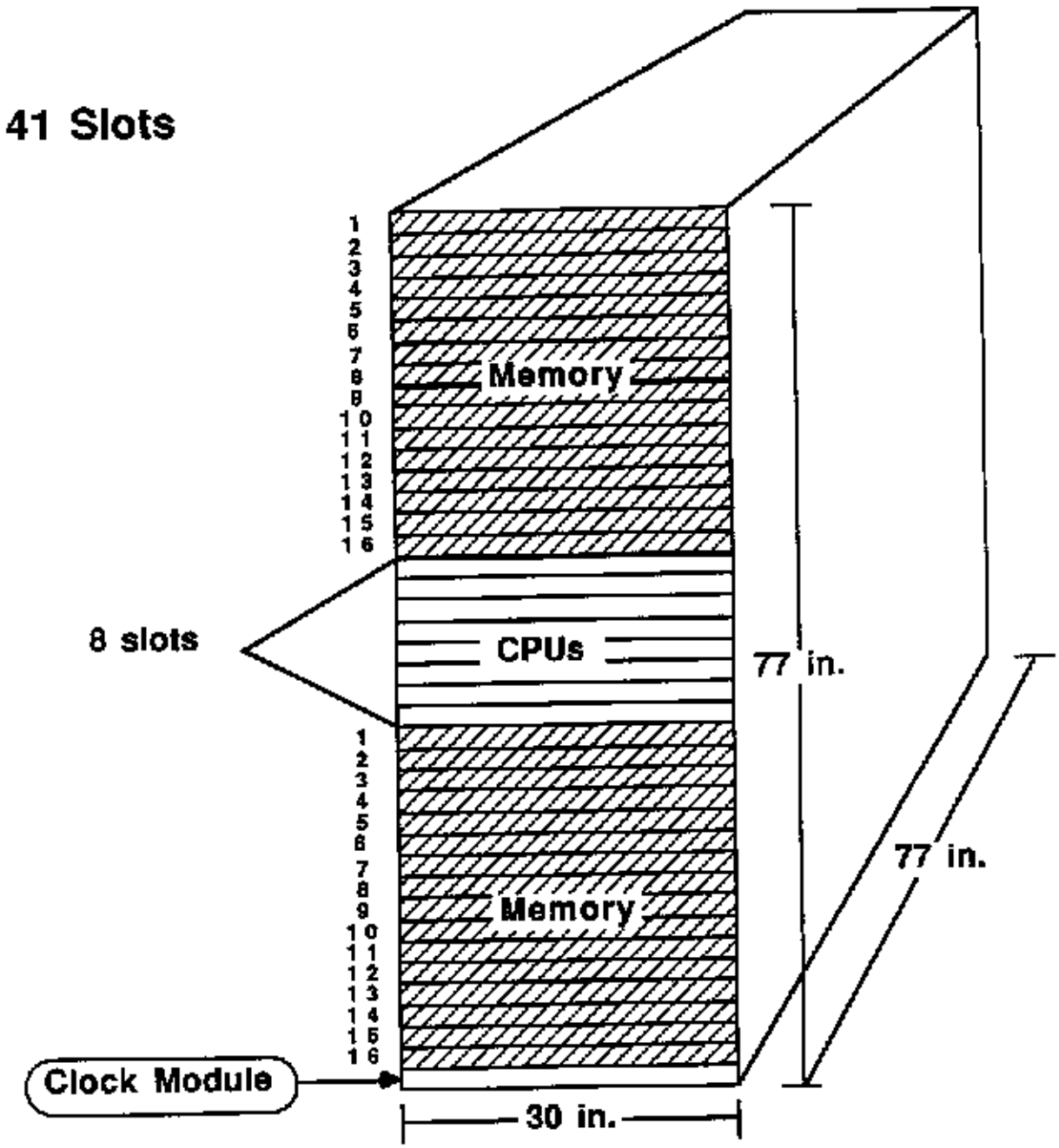
Y-MP4 CHASSIS DIMENSIONS



Y-MP8 CHASSIS COLUMN

- Total of 41slots in one column
  - 4 or 8 CPU modules
  - Up to 32 memory modules
  - 1 system clock module

Y-MP8 CHASSIS DIMENSIONS



Y-MP SHARED RESOURCES

CENTRAL MEMORY

• Word width	
- Data bits	64
- Error Correction bits	8
• Memory size (Mwords)	16, 32, 64, 128
• Number of banks (64 per section)	64, 128, 256
• Number of ports per CPU	4

I/O SECTION

• VHISP channels (1000 Mbytes/s)	1, 2
• HISP channels (100 Mbytes/s)	2, 4, 8
• LOSP channels (6 Mbytes/s)	2, 4, 8

SHARED REGISTERS, NINE CLUSTERS CONSISTING OF:

• Shared address (SB) 32 bits each	8
• Shared scalar (ST) 64 bits each	8
• Semaphore (SM) 1 bit each	32

REAL TIME CLOCK, 64 BITS 1

Y-MP8/832 PHYSICAL CHARACTERISTICS

(excluding attached IOS and SSD)

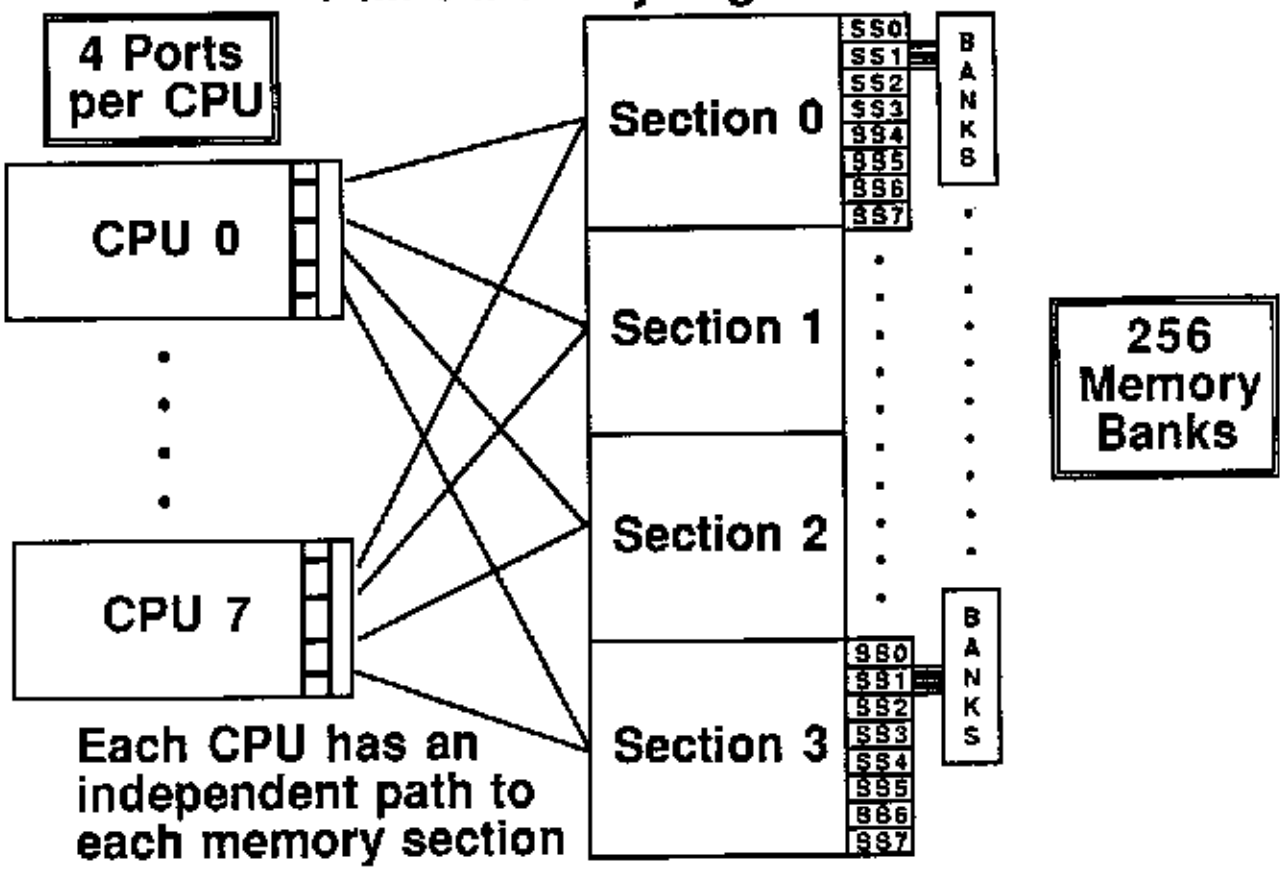
Floor space	18.4 square feet
Weight	5,000 pounds
Height	6.3 feet
Power	163 KVA
Cooling	80 gallons per minute Fluorinert (2 gallons per minute through each module) 10 degrees F. temperature difference in/out 42 tons cooled by chiller

CENTRAL MEMORY I/O PORTS

Machine	Ports/Processor	Total
Y-MP2/1	2 Read 2 Write 1 I/O	4
Y-MP2/2	2 Read 2 Write 1 I/O	8
Y-MP4	2 Read 2 Write 1 I/O	16
Y-MP8	2 Read 2 Write 1 I/O	32

- Shared Central Memory (16 Million to 128 Million 64-bit Words)
- 4 Memory Ports per CPU
- Each of eight CPUs can perform two reads, one write, and one I/O to central memory at the same time. At the rate of 4 64-bit words generated each clock period, peak memory bandwidth is 340 Gbits/s (8 CPUs * 4 words/cp * 64 bits/word)/6.0 ns = 340 Gbits/s (42.5 Gbytes/s) assuming no bank conflicts. This compares with 120 Gbits/s on an X-MP/48 system * 4 CPUs * 4 words/cp * 64 bits/word)/8.5 ns = 120 Gbits/s (15 Gbytes/s) assuming no bank conflicts.

Y-MP8 Memory Organization



CRAY Y-MP CONCEPTS

- CPU has scalar and vector instructions
- Eight vector registers per processor, each with 64 x 64-bit elements
- All arithmetic is register-to-register
- Eight independent functional units per processor can be used by vector instructions
- Pipelining:
    - All functional units are fully segmented; after initial startup, results are generated at the rate of one per clock period per processor
- Vector instructions using independent resources may execute in parallel
- Chaining:

  Result vector is eligible to become an operand once the first element has been computed

CPU COMPUTATION SECTION

Each CPU features:

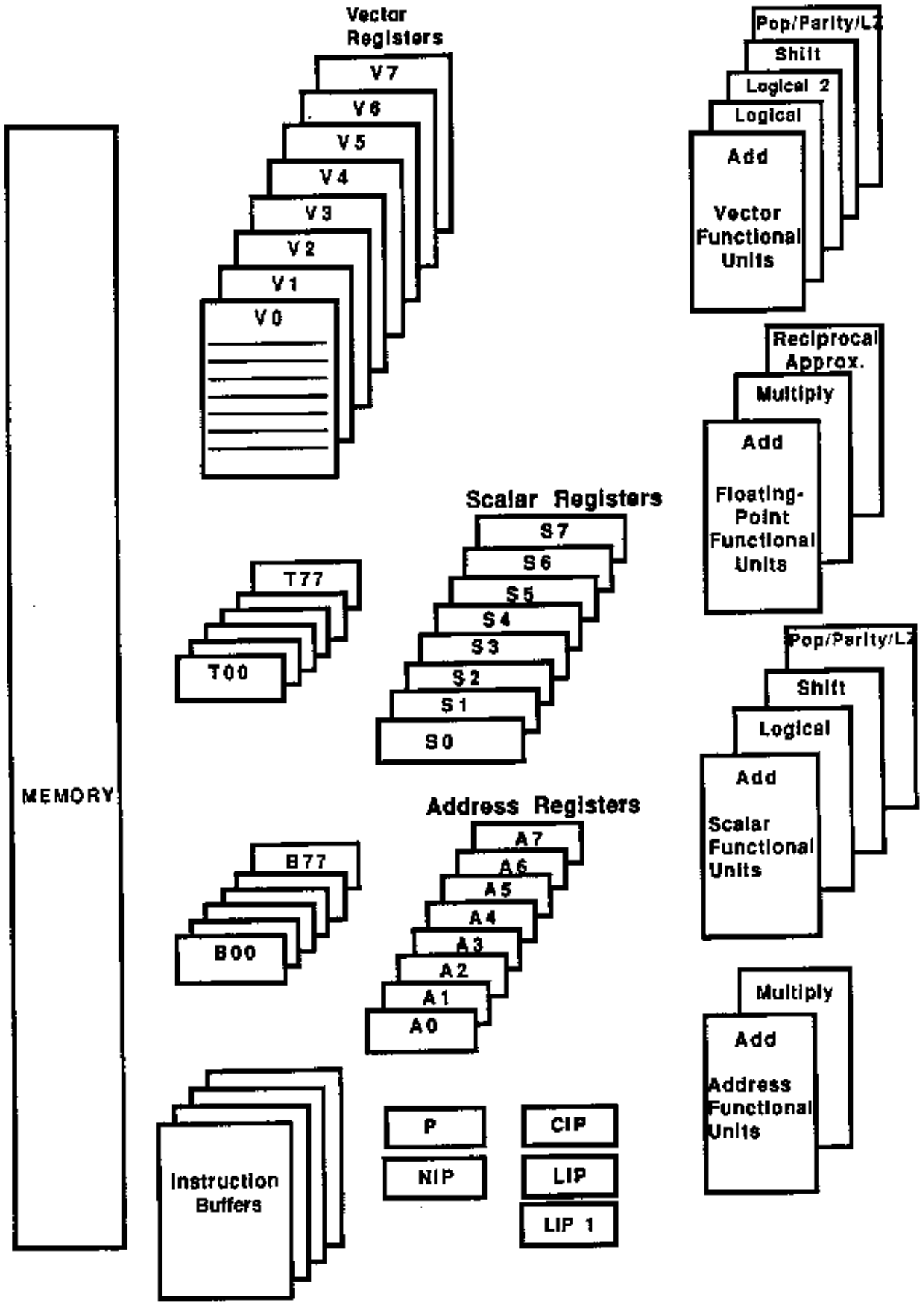
- 6.0 ns Clock Period
- 14 functional units
- 8 32-bit address registers (A registers)
- 64 32-bit intermediate address registers (B registers)
- 8 64-bit scalar registers
- 64 64-bit scalar save registers (T registers)
- 8 64-element (64-bit) vector registers
- Flexible chaining
- 64-bit floating-point arithemetic
- 64-bit integer arithemetic

CPU CONTROL SECTION

- Four Instruction Buffers Each Holding128 16-bit Instruction Parcels
- Instruction Buffers Loaded at 1 Word per Clock Period
- Normal and Interprocessor Interrupt Handling
- Constant Stride Addressing
- Gather/Scatter and Compress Index on all models
- Separate Program and Data Field Protection in Memory

CPU COMMUNICATION SECTION

- 9 Clusters of Intercommunications Registers
  - 8 Shared Address (32-bit)
  - 8 Shared Scalar (64-bit)
  - 32 Synchronization (1-bit) semaphores
- Monitor Mode Interprocessor Interrupts
- Hardware Deadlock Detection
- Real-time Clock
- Above hardware Shared by Multiple Processors



Cray Y-MP Simplified Block Diagram

FUNCTIONAL UNITS

The functional units in a Y-MP CPU are fully segmented to allow pipelining and chaining. That is, the specific operations required to complete an operation are broken down into one-clock period segments. This allows results to be generated at the rate of one per clock period per CPU. The types of functional units and their uses is as follows:

Address functional units

Addition	Performs 32-bit integer addition or subtraction; subtraction is accomplished by a process that essentially adds the negative of the number to be subtracted
Multiplication	Performs 32-bit integer multiplication

Scalar functional units

Addition	Performs 64-bit addition or subtraction; subtraction is accomplished in the same manner as an address add
Shift (single or double)	Shifts up to 128-bits (two concatenated S registers) to another S register resulting in a field of up to 64 bits
Logical	Performs bit-by-bit manipulation of 64-bit quantities from S registers; it is used to compare two sets of numbers for things like matches or differences
Pop/Parity/ Leading Zero	Counts the number of bits in an S register having a value of 1 (population count); it can also report if that sum is odd or even (parity); and it can count the number of leading bits (from left) which are equal to zero

Vector Functional Units

Addition	Performs 64-bit addition or subtraction as Scalar Addition, above, except each operand is just one element of a possible 64 elements in the V register
Shift	Performs a shift like a Scalar Shift, above, but each shift is on one or two elements of a possible 64 elements within the V register
Logical	Performs logical compares like Scalar Logical, above, but works with up to 64 64-bit elements in the V register; there are two vector logical functional units. Due to space constraints, the second vector logical unit uses part of the Floating-point Multiply unit and, thus, has takes longer than the Full Vector Logical unit
Pop,parity	Performs population counts and parity checks like scalar, above, but works on elements within a vector register; leading zero counting is not performed in this functional unit

Floating Point functional units (S and V registers only)

Addition	Performs 64-bit addition or subtraction on very large or very small numbers and can represent them using exponential notation and decimal points
Multiplication	Performs 64-bit multiplication on the same types of numbers as in floating-point addition; number rounding is available with this functional unit
Reciprocal Approximation	Performs an equivalent to a floating-point division by utilizing Newton's method of a series of reciprocal multiplications to achieve an "approximate" answer

TABLE OF FUNCTIONAL UNITS

UNIT	REGISTER	CLOCK PERIODS
ADDRESS FUNCTIONAL UNITS		
ADDITION	A	2
MULTIPLICATION	A	4
SCALAR FUNCTIONAL UNITS		
ADDITION	S	3
SHIFT		
SINGLE	S	2
DOUBLE	S	3
LOGICAL	S	1
POPULATION, PARITY AND LEADING ZERO	S	3 or 4
VECTOR FUNCTIONAL UNITS		
ADDITION	V	3
SHIFT	V	3 or 4
FULL VECTOR LOGICAL	V	2
2ND VECTOR LOGICAL	V	4
POPULATION, PARITY	V	5
FLOATING-POINT FUNCTIONAL UNITS		
ADDITION	S AND V	6
MULTIPLICATION	S AND V	7
RECIPROCAL APPROX.	S AND V	14

IOS OPTIONS ON THE Y-MP

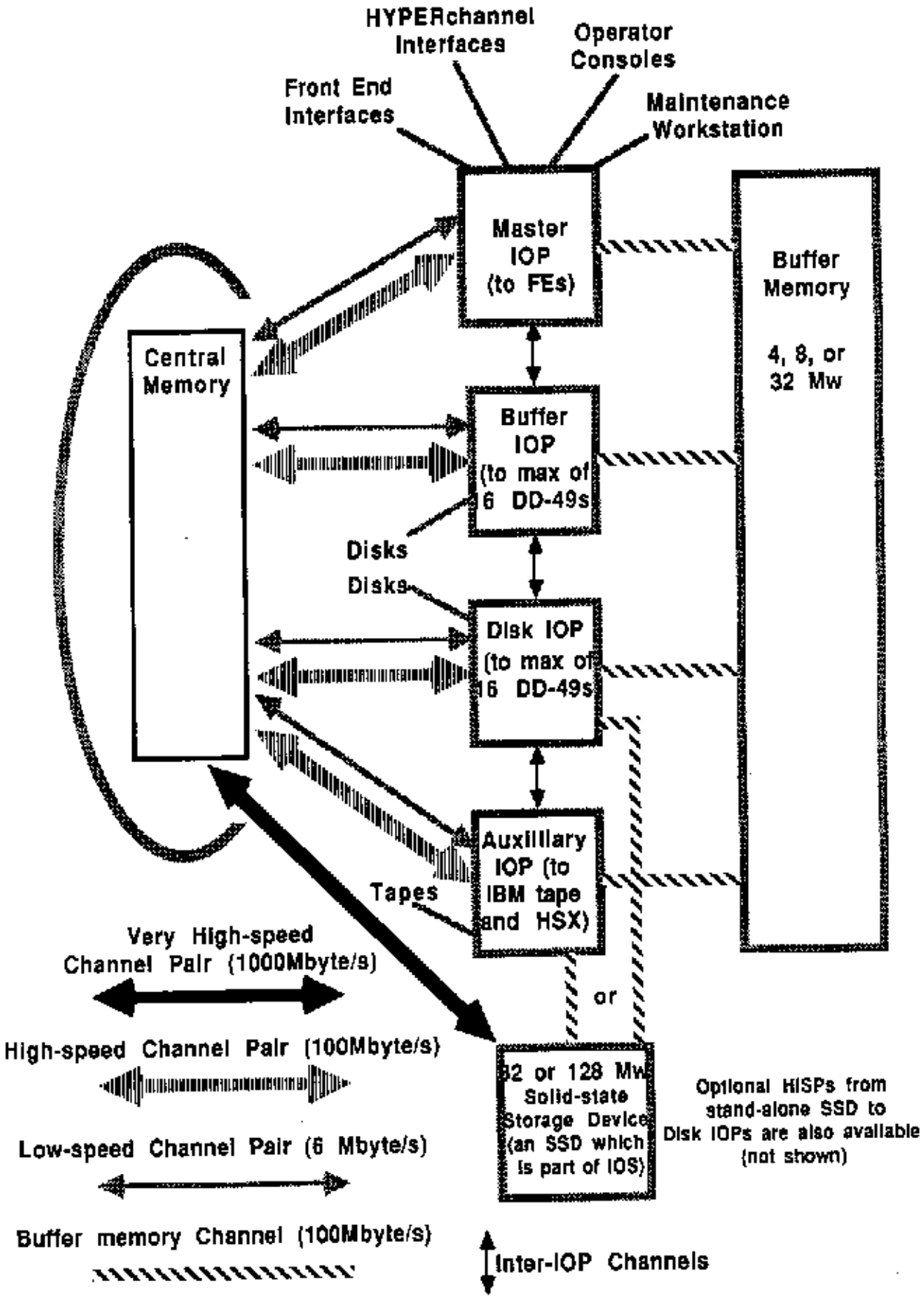
The Model D I/O Subsystem enhances the Y-MP models to enable fast, efficient data access and processing.

- Y-MP2 and Y-MP4 IOS configuration options include:
    - one standard I/O subsystem containing an MIOP and a BIOP with expansion to include a DIOP and/or an XIOP
    - one standard LSP-4 in the MIOP (which allows connections to the Cray mainframe and up to three front-end interfaces or Network Systems Corp. adapters) with expansion to include an optional LSP-4 (which adds four more FEI/NSC adapter connections)
    - the BIOP and DIOP can hold up to three DCU-5 disk controllers each and each of them can drive four disk drives
    - the XIOP can contain one HSX-1 (high-speed external channel connection) and up to two BMC-5 tape controllers which can support up to eight IBM tape channels
    - 4 Mwords standard buffer memory, with options to expand to 8 or 32 Mwords

The IOS attaches to one side of the Y-MP4 as a "wing". The IOS on a Y-MP2 is situated in front facing the mainframe.

- Y-MP8 IOS configuration options include:
    - one standard I/O subsystem containing an MIOP, a BIOP, a DIOP, and an XIOP; this IOS forms a "wing" on the mainframe
    - all of the above mentioned expansion options
    - the addition of a second IOS which faces the mainframe

I/O Subsystem - Model D



SOLID-STATE STORAGE DEVICE (SSD) OPTIONS

- The SSD functions as a very high speed secondary memory and is looked at by the operating system and user as another disk drive

- The current memory size options are 128, 256, or 512 Mwords (a 32 or 128 Mword SSD can optionally be located in the I/O subsystem; this is the only option with the Y-MP2)

- The user accesses the SSD via Cray job statements with no changes to the program necessary

- Transfer rates are from 100 to 2000 Mbytes/second; access time is less than 25 microseconds

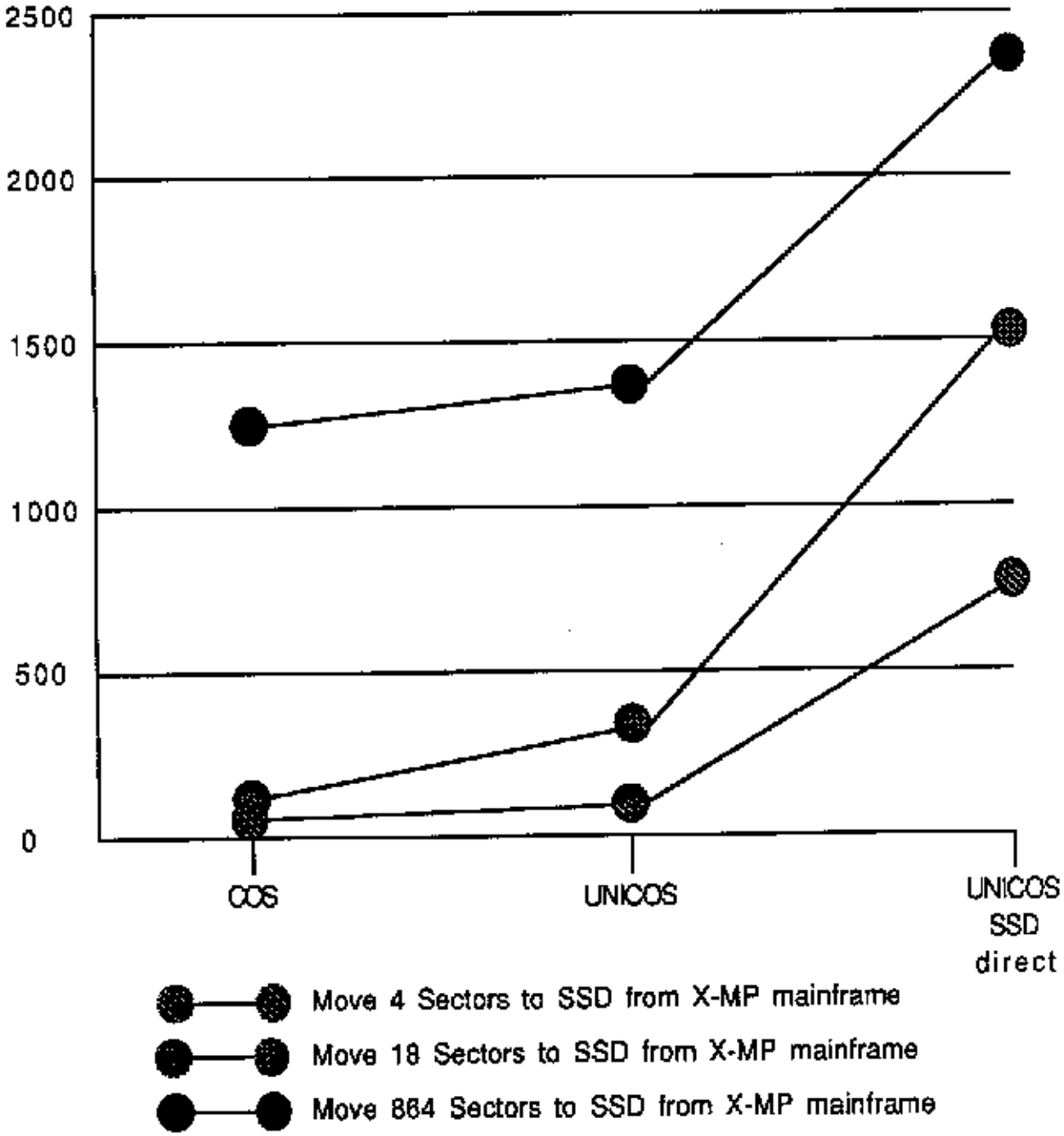
- Connects to:

  CRAY Y-MP4 via one or two 1000 Mbytes/s channel
  CRAY Y-MP8 via one or two 1000 Mbytes/s channels
  (The optional 32 or 128 Mword SSD located in the I/O Subsystem is connected via one 1000 Mbytes/s channel.)

- An optional SSD direct connection to the IOS is available which can allow faster throughput of disk data to CRAY central memory

- Provides a significant performance improvement on I/O-bound programs

SSD PERFORMANCE



CRAY Y-MP DISK DRIVE OPTIONS

CRAY Y-MP computers support:

- DS-40 disk subsystems
  - 5.2 Gbytes capacity per DD-40 disk storage unit
  - 20.8 Gbytes capacity per DS-40 (5.2 Gbytes x 4 DSUs/subsystem)
  - 9.6 Mbyte/s sustained transfer rate for each DD-40 disk unit
  - 16 ms average access time
  - DCU-5 controller
- DD-49 disk drives
  - 1.2 Gbytes capacity
  - 9.6 Mbyte/s sustained transfer rate
  - 16 ms average access time
  - DCU-5 controller
- RDS-5 Disk System
  - 4 Removable HDA Assemblies
  - 1200 Mbyte capacity each
  - 3.0 Mbyte/s transfer rate

DISK STRIPING

- Software technique which allows an increase in disk transfer rate (up to a factor of 15 with a second IOS and UNICOS 5.0)
- Multiple disks are treated as one logical device
- Striped groups are limited to like disk types
- Access via a single command
- Simultaneous I/O on all disks in the group
- Effective I/O rate is multiplied by the number of disks in the group

History of Cray Disk Storage Devices

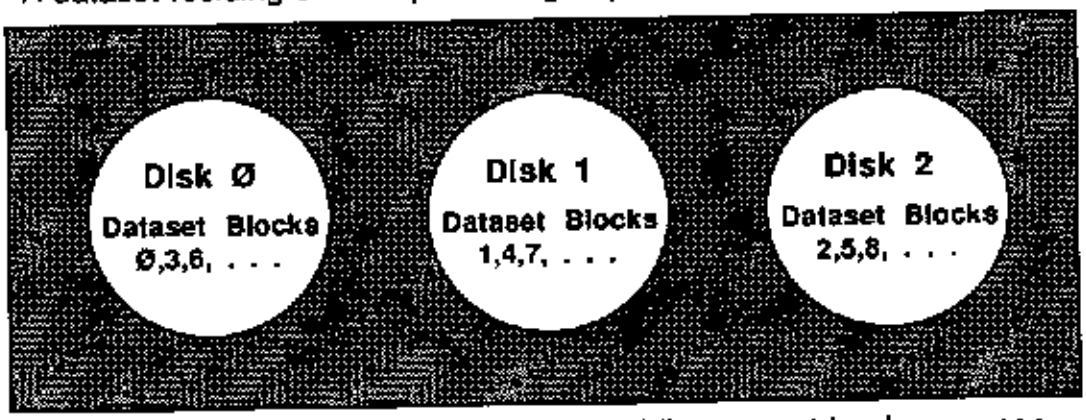
Discontinued | Current Products

Model	DD-19	DD-29	DD-39	DD-49	DD-40
Manufacturer	CDC	CDC	Fujitsu	Ibis	CDC
Storage Capacity	303 Mb	606 Mb	1238 Mb	1219 Mb	5200 Mb
Transfer Rate (max)	32 Mbits/s	32 Mbits/s	52 Mbits/s	82 Mbits/s	82 Mbits/s
Drives per Device	1	1	3	1	4
Cray systems	1,X	1,X,2	X,Y	X,2,Y	X,2,Y

Disk Striping

Disk Striping allows a group of disks to be viewed by the operating system and user as a single logical device, but which yields increased I/O performance.

A dataset residing on a striped disk group looks like this:



A read or write request to a dataset residing on a striped group can cause simultaneous I/O to occur on all disks in the group.

In effect, the I/O rate for the dataset is multiplied by the number of disks in the group.

All disks in the striped group must be the same type and must be connected to the I/O Subsystem.

DAISY CHAINING

The DS-40 disk subsystem includes a facility to effectively allow doubling the storage capacity. This is known as daisy chaining.

Because each DC-40 disk controller in the DCC-2 chassis contains a dual-ported interface, as many as eight DSUs can be connected to the DCC-2 chassis. Only four DSUs can be active at one time, however.

The effect of daisy chaining a single subsystem doubles capacity from 21 Gbytes to 42 Gbytes (4 x 5200 Mbytes vs. 8 x 5200 Mbytes).

Note that this does not mean twice the performance since the chained devices share a single multiplexed channel to the DCU-5 in the IOS.

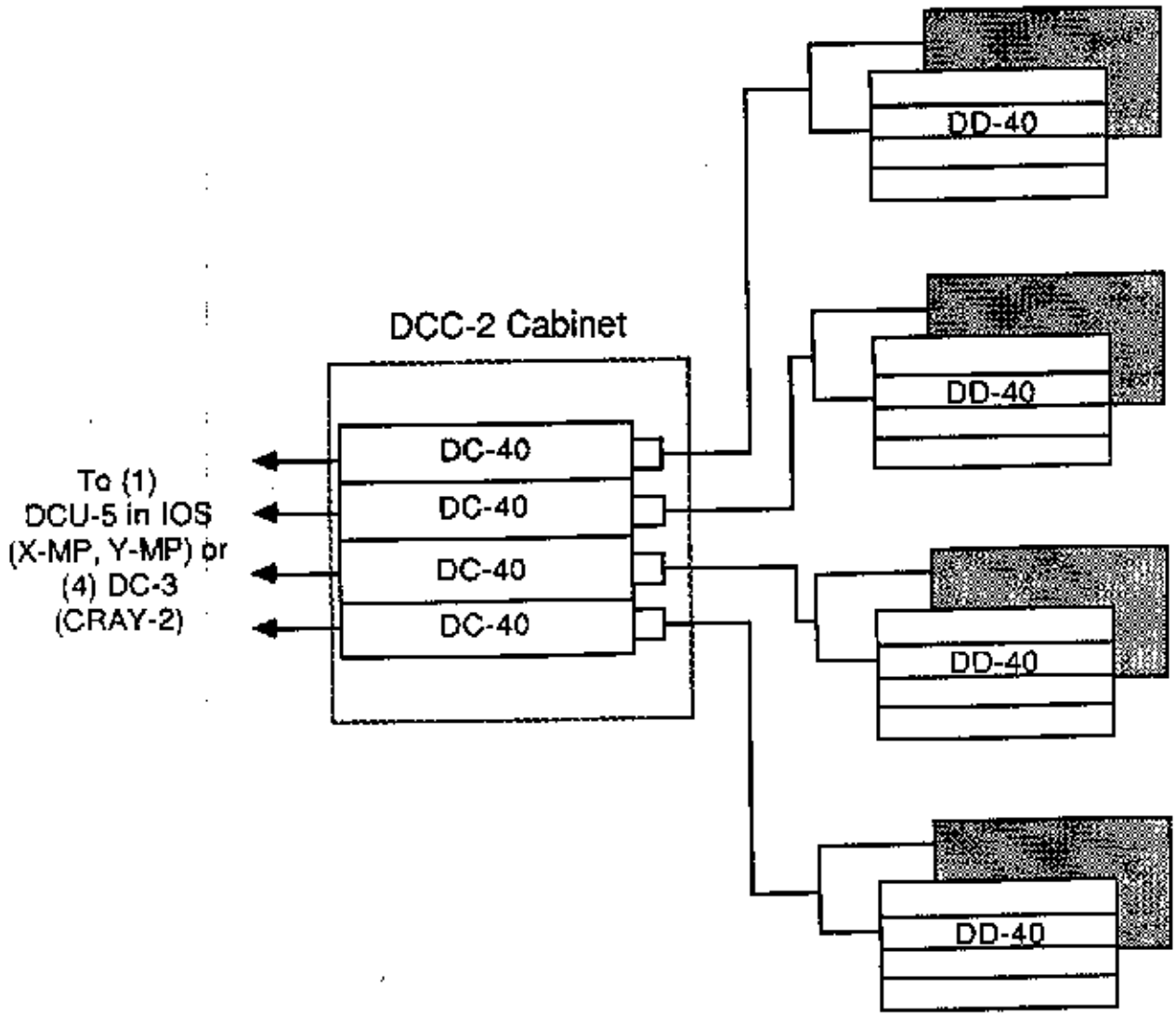
A Y-MP8 with two IOS Model Ds could have up to 48 disk storage units. Up to 96 disk storage units are possible with DS-40 daisy chaining yielding 249.6 Gbytes (499.2 Gbytes with daisy chaining).

DS-40 OPTIONS

			Capacity	
CRAY Model	IOS	DS-40 Subsystems	Without daisy chaining	With daisy chaining
Y-MP	1 Model D	6 (24 DSUs)	124.8	249.6
Y-MP	2 Model D	12 (48 DSUs)	249.6	499.2

Capacity values are in Gbytes.

DS-40 DISK SUBSYSTEM
with daisy chaining

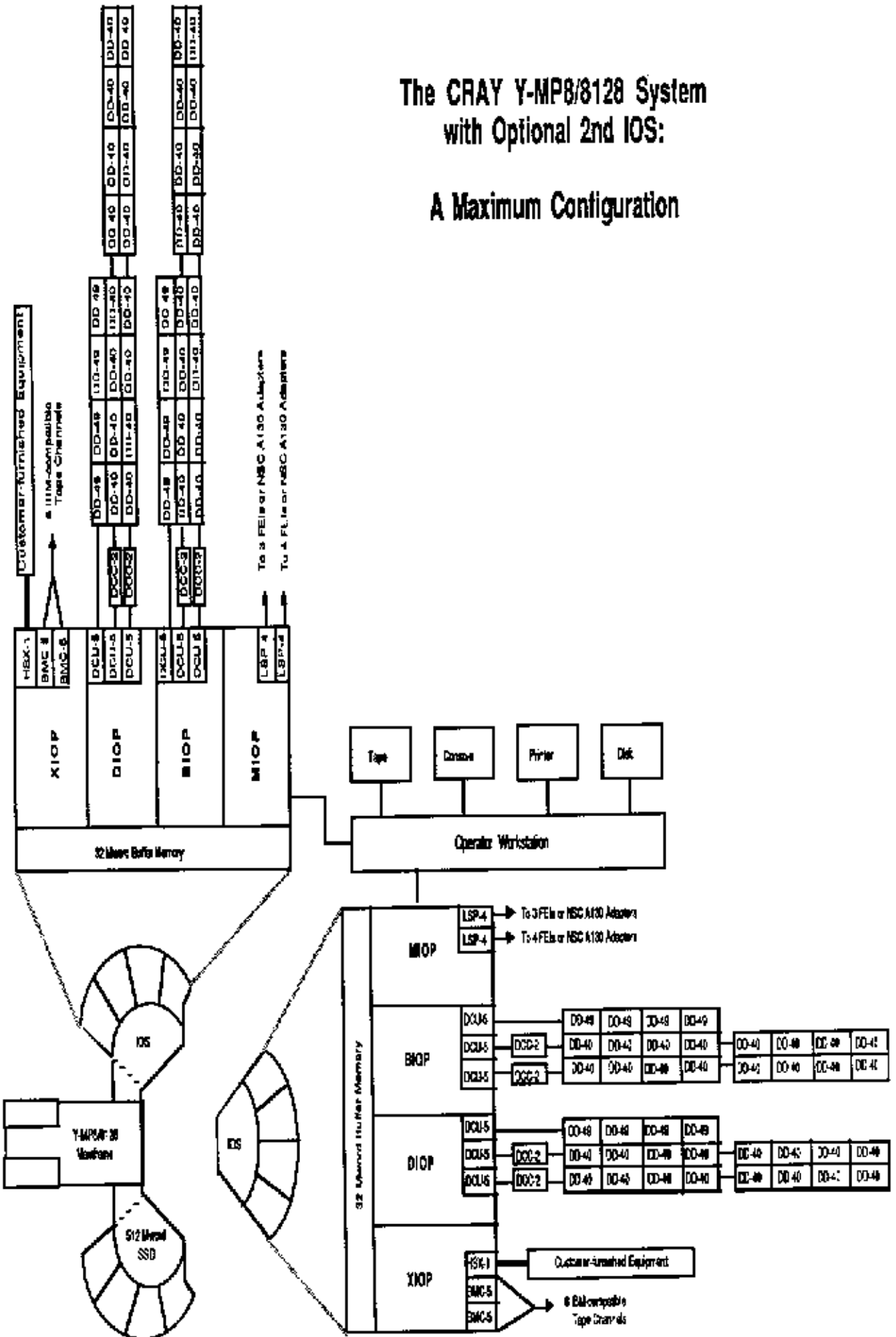


CRAY Y-MP and CRAY X-MP COMPARISON

Characteristic	CRAY Y-MP Mainframe	CRAY X-MP EA Mainframe
Memory size Banks	16, 32, 64, 128 Mwords 64, 128, 256	4, 16, 32, 64 Mwords 16, 32, 64
CPUs	1, 2, 4, 8	1, 2, 4
A register width	24 bits (X-mode) 32 bits (Y-mode)	24 bits (X-mode) 32 bits (Y-mode)
Address register width	24 bits (X-mode) 32 bits (Y-mode)	24 bits (X-mode) 32 bits (Y-mode)
SB registers per cluster	8 x 24 (X-mode) 8 x 32 (Y-mode)	8 x 24 (X-mode) 8 x 32 (Y-mode)
Clusters	0, 3, 5, 9	0, 3, 5
Instruction formats	1 and 2 parcels (X-mode) 1,2, and 3 parcels (Y-mode)	1 and 2 parcels (X-mode) 1,2, and 3 parcels (Y-mode)
Instruction expression (maximum width) exp =	24 bits (X-mode) 32 bits (Y-mode) *ijkm* (X-mode) *nm* (Y-mode)	24 bits (X-mode) 32 bits (Y-mode) *ijkm* (X-mode) *nm* (Y-mode)
Program block Maximum size Boundary	 4 Mwords 256 words	 4 Mwords 256 words
I/O channels 6 Mbyte/second 100 Mbyte/second 1000 Mbyte/second	 2, 4, 8 2, 4, 8 1, 2	 1, 2, 3, 4 1, 2, 3, 4 1, 2
I/O subsystems Maximum IOPs	2 (1 is std., 1 more opt. on Y-MP8) (4 is std, 4 more opt.)	1 4
Maint. workstation	VMEbus system	VMEbus system
Error Logger	VMEbus system	VMEbus system
Operator workstation	VMEbus system	VMEbus system
Disk drives	DD-39s, DD-40s, DD-49s	DD-39s, DD-40s, DD-49s

The CRAY Y-MP8/8128 System with Optional 2nd IOS:

A Maximum Configuration



CRAY Y-MP PURCHASE PRICES
(Made available by instructor)

SECTION 4

Cray Product Familiarization

CRAY X-MP EA

Computer System

Cray Research, Inc. Software Training

THIS PAGE IS INTENTIONALLY LEFT BLANK.

CRAY X-MP EA SYSTEMS

The X-MP series of computers has been a mainstay of the Cray supercomputer product line since its introduction in 1982. Now, however, with the introduction of the new Y-MP models, most of the X-MP EA (as it is now known) models are being discontinued. The only X-MP EA models still being produced will be the "se" models described later in this section.

Below is a recap of the X-MP EA models that have been and will still be available along with some physical characteristics of them.

CRAY X-MP EA SERIES OF SYSTEMS

Model	CPUs	Memory Size (Mwords)†	Banks	IOPs	Buffer Memory (Mwords)†
X-MP EA/14se	1	4	16	2/3	2/4/8/32
X-MP EA/18se	1	8	16	2/3	2/4/8/32
X-MP EA/116se	1	16	16	2/3	2/4/8/32
X-MP EA/116 *	1	16	32	2/3/4	4/8/32
X-MP EA/132 *	1	32	32	2/3/4	4/8/32
X-MP EA/164 *	1	64	64	2/3/4	4/8/32
X-MP EA/216 *	2	16	32	2/3/4	4/8/32
X-MP EA/232 *	2	32	32	2/3/4	4/8/32
X-MP EA/264 *	2	64	64	2/3/4	4/8/32
X-MP EA/416 *	4	16	32	2/3/4	4/8/32
X-MP EA/432 *	4	32	32	2/3/4	4/8/32
X-MP EA/464 *	4	64	64	2/3/4	4/8/32

IOS
SSD

* Out of production

† 1 Mword = 1,048,576 words

PHYSICAL CHARACTERISTICS

Model	No. of Columns	Arc in Degrees	Sq.Ft.	Tons	KVA Power
X-MP EA/1se	6	135	21	3.0	80
X-MP EA/1	8	182	26	3.8	80
X-MP EA/2	8	182	26	3.8	80
X-MP EA/4	12	270	39	5.7	150

IOS weighs 1.7 tons, occupies 13 sq.ft.
SSD weighs 1.7 tons, occupies 13 sq.ft.

CRAY X-MP EA/se SYSTEMS OVERVIEW

Three models of the X-MP EA/se are now available and are the only X-MP systems still in production.

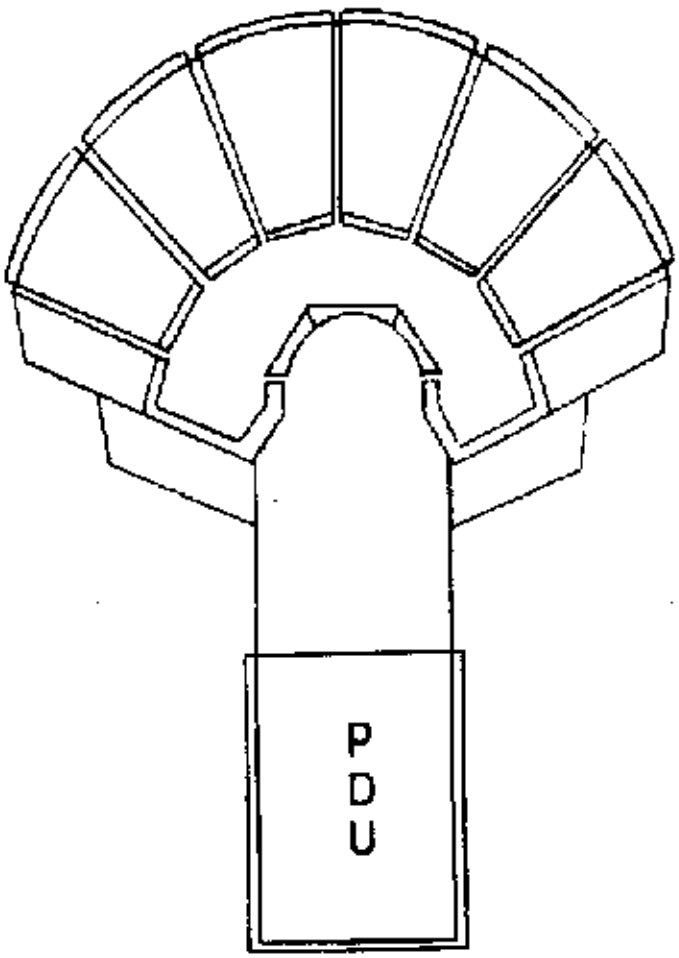
Each model is a complete system in six columns and contains one CPU. The only options are in the amount of central memory and the IOS options. The models are:

- X-MP EA/14se (4 Mwords of Static MOS memory in 16 banks)
- X-MP EA/18se (8 Mwords of Static MOS memory in 16 banks)
- X-MP EA/116se (16 Mwords of Static MOS memory in 16 banks)

The clock rate of the X-MP EA/se is 8.5 ns. Earlier versions with a 10.0 ns clock rate can be upgraded.

The X-MP EA/se can operate in either X-MP or Y-MP mode.

CRAY X-MP EA/se



6-columns
Internal IOS
PDU attached

X-MP EA/se IOS CONFIGURATION OPTIONS

The standard built-in IOS system includes:

- An MIOP
- A BIOP
- Two Mword buffer memory
- One 100 Mbyte/sec. channel
- One 6 Mbyte/sec channel
- One LSP-4 for connecting to front-ends and networks
- One DCU-5 for connecting to disk units

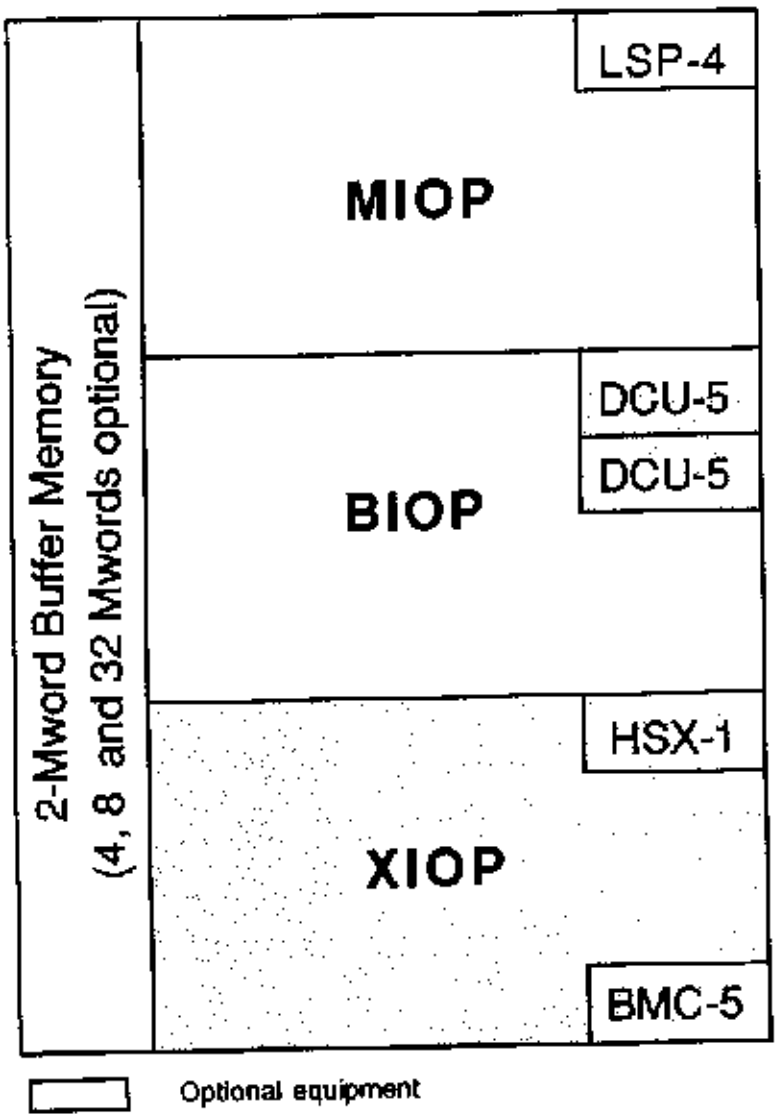
Options include:

- Memory upgrades to 4, 8, or 32 Mwords
- An XIOP
- A BMC-5 for connecting to IBM tape equipment (requires an XIOP)
- One more DCU-5

No SSD options are available with the X-MP EA/se systems.

The X-MP EA/se supports the DD-49 disk units and the DS-40 disk subsystem.

CRAY X-MP EA/se IOS Diagram



CRAY X-MP EA/se PURCHASE PRICES
(Made available by instructor)

SECTION 5

Cray Product Familiarization

CRAY-2

Computer System

Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK.

CRAY-2 SUPERCOMPUTING PHILOSOPHY

- Design simplicity
- Few, very fast processors
  - High-speed logic circuits
  - Register-based
  - Fast scalar processing
  - Fast vector processing
- Parallel processing
  - Vector operations
  - Separate functional units
  - Multiple processors and multitasking
- Very high performance I/O
- Very large common memory
  - Highly interleaved memory
  - Accessible from each processor
- High availability
- Ease of system integration
  - Networks and stations
- Standard software
  - Fortran
  - Operating system based on AT&T UNIX* System V

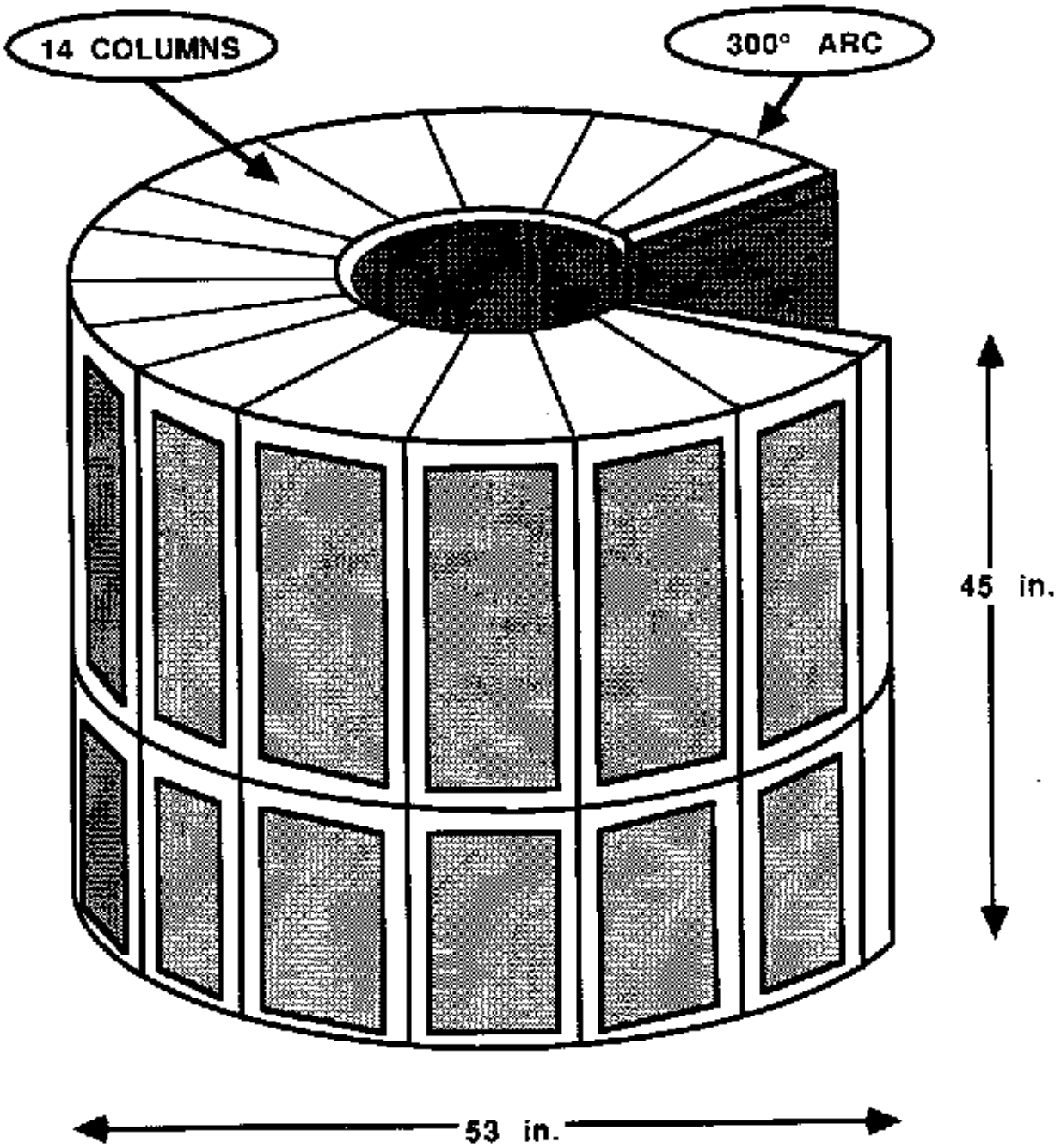
*UNIX is a trademark of AT&T Bell Laboratories

PHYSICAL APPEARANCE

- SMALL C-SHAPED CABINET CONTAINING 14 COLUMNS IN A 300° ARC
- COLUMNS ARRANGED IN A PARTIAL CIRCLE WITH A 20° ANGLE BETWEEN COLUMNS
- MODULES COMPRISE THE UPPER 24 INCHES OF EACH COLUMN
- POWER SUPPLIES COMPRISE THE LOWER 19 INCHES OF EACH COLUMN

PHYSICAL CHARACTERISTICS

- 16 SQUARE FEET FLOOR SPACE
- 45 INCHES HIGH
- 53 INCHES DIAMETER
- 2.75 TONS
  - 5500 POUNDS FILLED
  - 3495 POUNDS DRY
  - (2005 POUNDS WEIGHT OF FLUORINERT)
- 2.96 POUNDS PER SQUARE INCH PRESSURE
- 195 KILOWATTS
- 14 COLUMNS IN 300° ARC
  - 336 MODULES TOTAL
- 36,000 TWISTED PAIR WIRES
  - 10 INCHES AVERAGE LENGTH
  - 25 INCHES LONGEST
  - 2.5 INCHES SHORTEST
  - 5.68 MILES


14 COLUMNS
300° ARC
45 in.
53 in.

COOLING

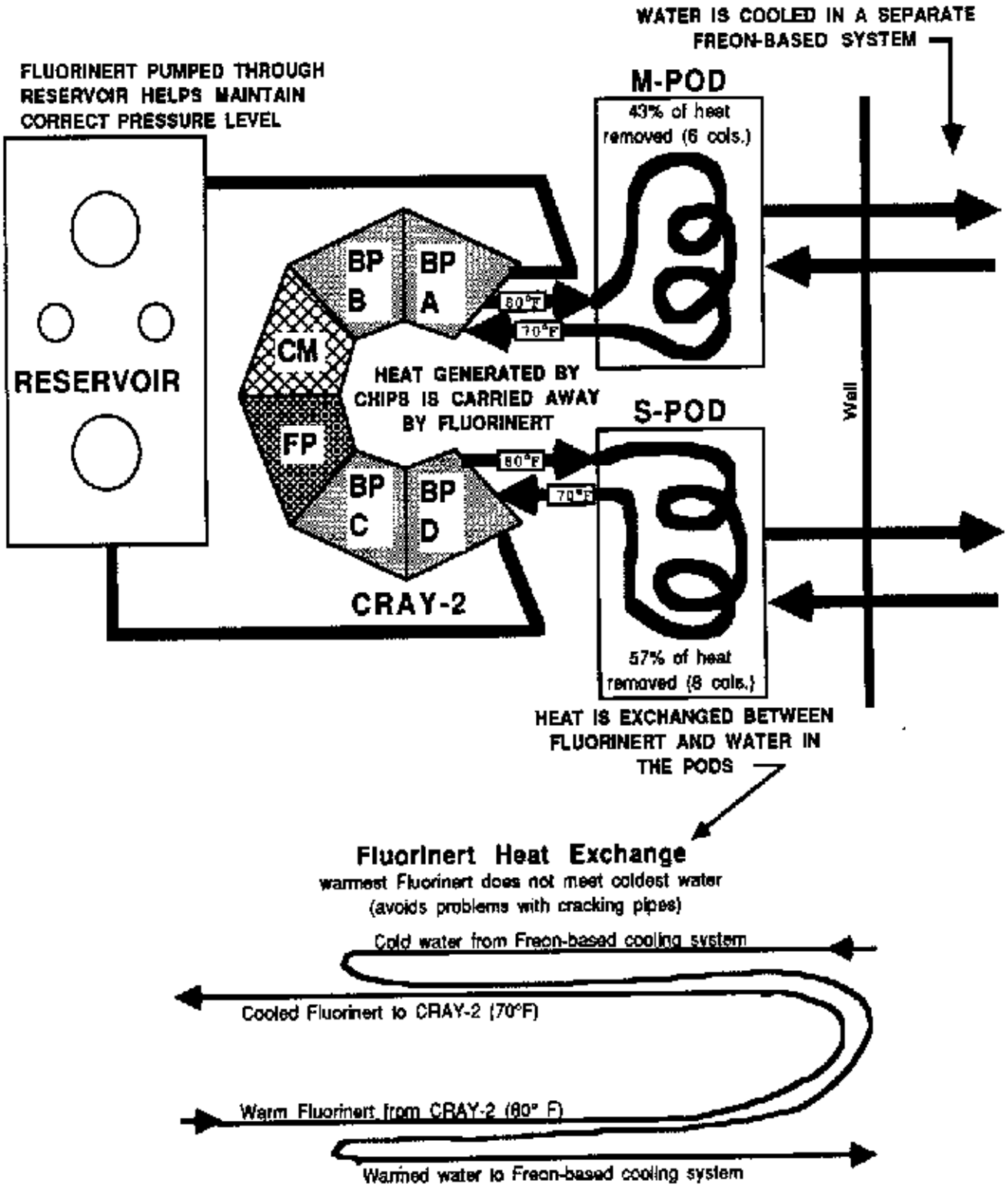
- LIQUID IMMERSION COOLING ALLOWS DENSELY-PACKED ELECTRONICS

- "VALVELESS" COOLING SYSTEM
  - FLUOROCARBON FLUID - 200 GALLONS
  - 1 INCH/S FLOW RATE
  - 70° F (21° C) CHASSIS ENTRY
  - 80° F (27° C) CHASSIS EXIT
  - STAND PIPE AND RESERVOIR
  - SHELL AND TUBE HEAT EXCHANGE

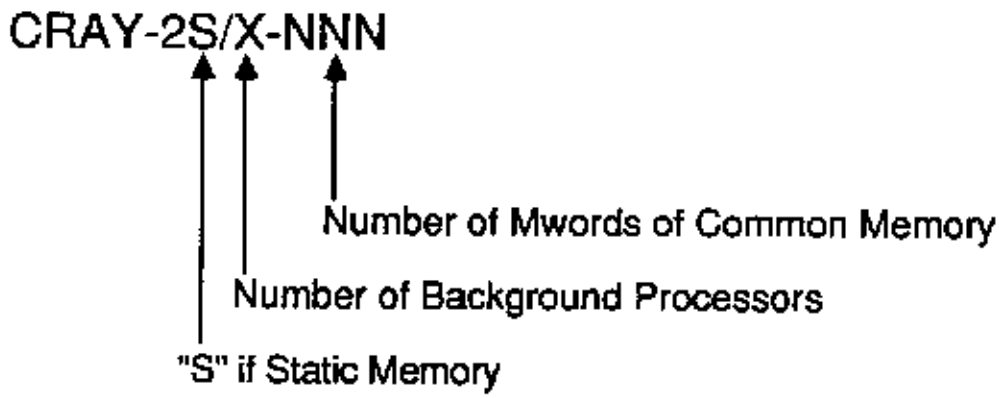
- FLUOROCARBON FLUID (PRIMARY)
  - COLORLESS AND ODORLESS
  - INERT AND NONTOXIC
  - NONFLAMMABLE AND INSULATING
  - HIGH THERMAL STABILITY
  - HIGH HEAT TRANSFER

- CHILLED WATER COOLING (SECONDARY)
  - 55 TONS
  - 50° F (10° C) NORMAL

CRAY-2 TWO-PHASE COOLING SYSTEM UTILIZING A DIELECTRIC SOLUTION (FLUORINERT), WATER, AND FREON



CRAY-2 MODEL NUMBERING SCHEME

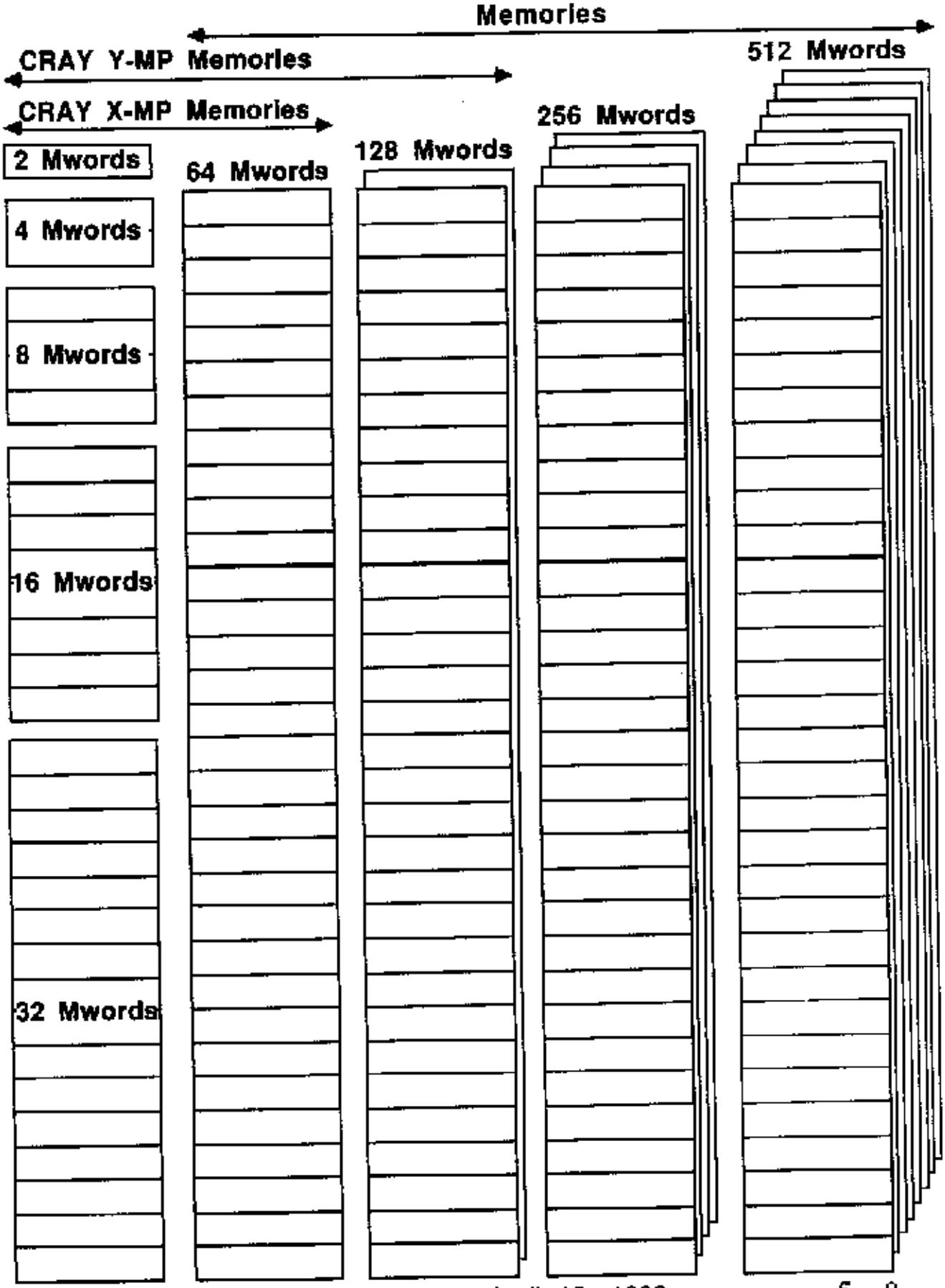


Dynamic Memory Model	Static Memory Models
CRAY-2/4-256	CRAY-2S/2-64
CRAY-2/4-512	CRAY-2S/2-128
	CRAY-2S/4-128

CRAY-2 OPTIONS

	CRAY-2 Model	Number of Background Processors	Common Memory (Mwords)	Number of Memory Banks	Mwords per Bank	Number of I/O Ports
DYNAMIC	4-256	4	256	128	2	40
	4-512	4	512	128	4	40
STATIC	2-64	2	64	64	1	20
	2-128	2	128	128	1	20
	4-128	4	128	128	1	40

Memory Size Comparison



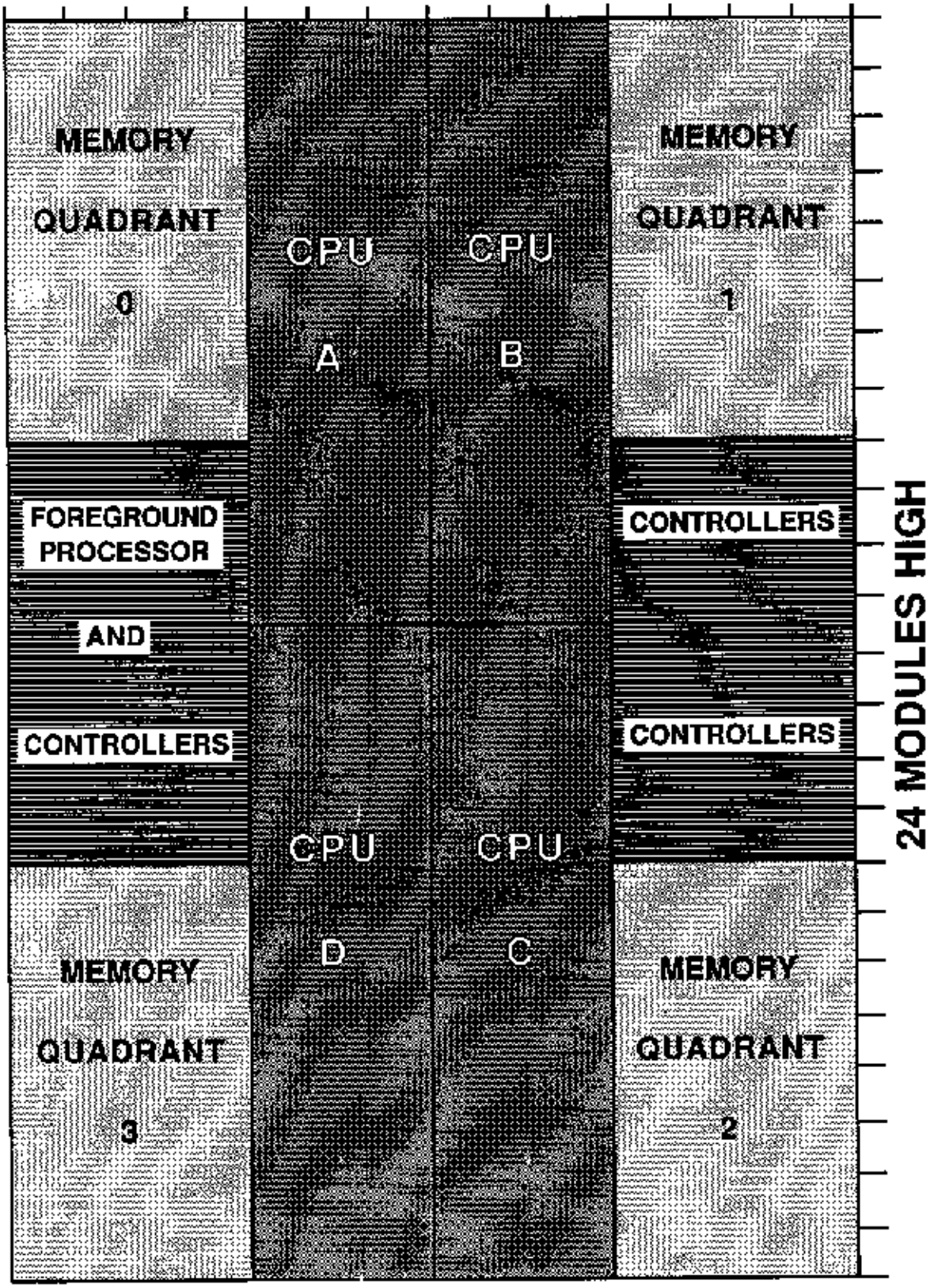
MODULE TECHNOLOGY

- THREE-DIMENSIONAL ORGANIZATION
  - 4x8x1 INCHES (10x20x3 CM)
  - 3-D CIRCUIT CONNECTIONS
  - 8-8x12 IC ARRAYS PER MODULE
  - 2 POUNDS PER MODULE
  - 300-500 WATTS PER MODULE
  - 8x36 PIN CONNECTORS (288 PER MODULE)
  - 750 ICs PER MODULE
- 64 OR 128 MEMORY MODULES
- 38 MODULES PER CPU
- THREE FOREGROUND PROCESSOR MODULES
- UP TO 40 I/O CONTROLLER MODULES
  I/O MODULES PROVIDE A CONNECTION TO A COMMUNICATION CHANNEL OR A CONNECTION TO DISK OR TAPE EQUIPMENT
- 36 MODULE TYPES

IC TECHNOLOGY

- FAST 16-GATE ARRAY LOGIC CHIPS
- 256Kbit or 1 Mbit MEMORY CHIPS
- UP TO 72,000 MEMORY CHIPS
- UP TO 240,000 ICs TOTAL

MODULE LAYOUT (4 processor system)



COMMON MEMORY

- 64, 128, 256, or 512 MWORDS
- 64 DATA BITS, 8 SECDED BITS
- 64 or 128 BANKS - 1 or 2 MWORD/BANK
- 256K MOS MEMORY TECHNOLOGY
- TOTAL MEMORY BANDWIDTH 64 GBITS
  FOUR 16 GBIT/s PORTS
- 4 COMMON MEMORY PORTS USED FOR:
  BACKGROUND PROCESSOR OPERAND REQUEST
  BACKGROUND PROCESSOR INSTRUCTION FETCH
  FOREGROUND PROCESSOR TRANSFER REQUEST
  I/O CONTROLLER TRANSFER REQUESTS

MEMORY TYPES

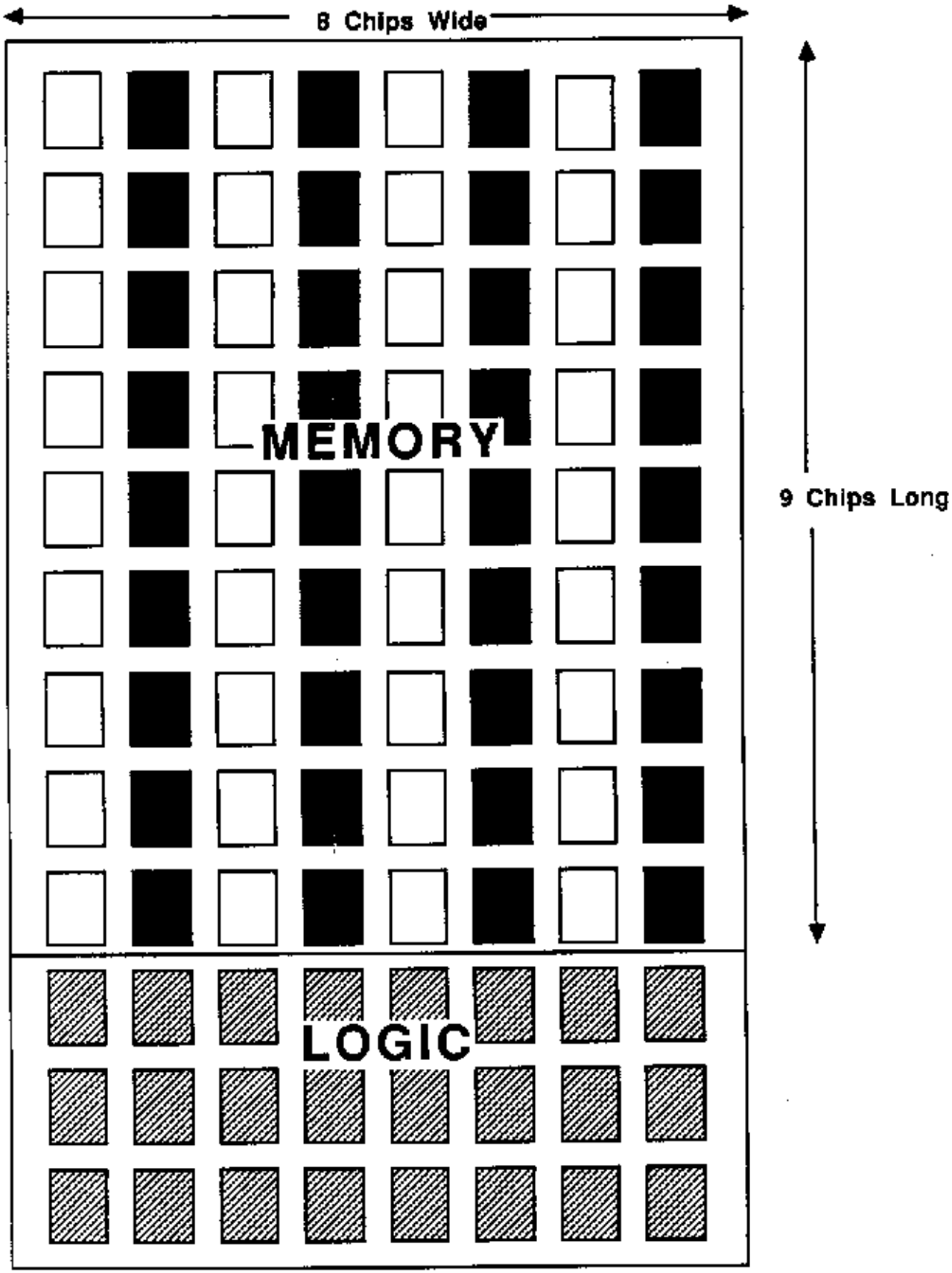
DYNAMIC RAM

- 256Kbit or 1 Mbit MOS
- CHIP ACCESS TIME: 80ns (120ns ON FIRST SYSTEMS)
- LONG BANK BUSY TIME: 45 CLOCK PERIODS
- SHORT BANK BUSY TIME: 13 CLOCK PERIODS
- MEMORY ACCESS TIME IS 47 CLOCK PERIODS (8 CPs FASTER THAN 120 ns DRAM)
- MAXIMUM MACHINE SIZE: 512 MWORDS

STATIC RAM

- 256K-BIT MOS
- CHIP ACCESS TIME: 45 ns
- BANK BUSY TIME: less than16 CLOCK PERIODS
- MEMORY ACCESS TIME: 35 CLOCK PERIODS (20 CPs FASTER THAN 120 ns DRAM)
- MAXIMUM MACHINE SIZE: 128 MWORDS

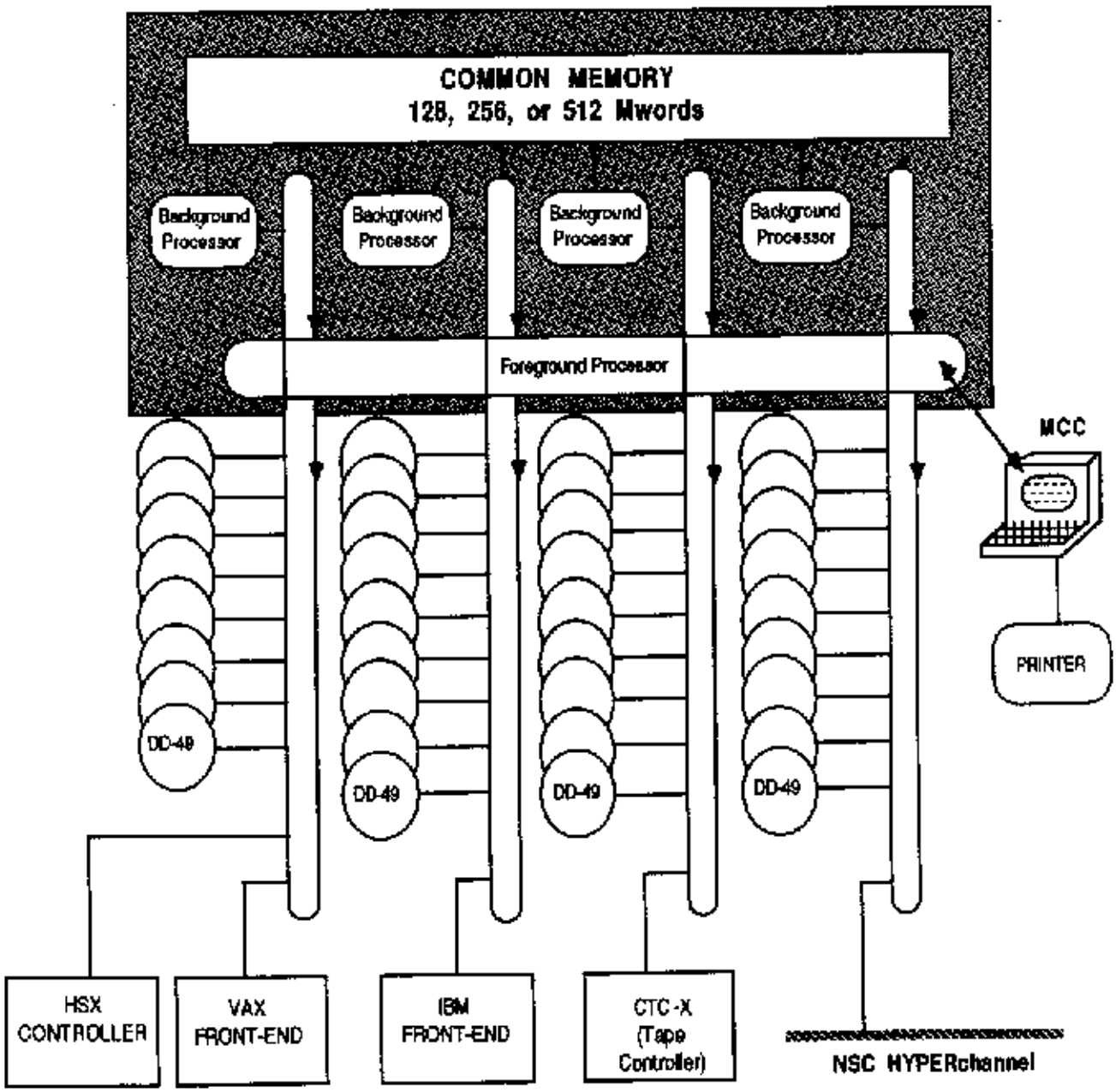
BOARD LAYOUT



CRAY-2 FOUR CPU ARCHITECTURE

- FOUR IDENTICAL BACKGROUND PROCESSORS
  - 16K 64-BIT WORDS LOCAL MEMORY PER PROCESSOR
  - SCALAR PROCESSING
  - VECTOR PROCESSING
  - INTERPROCESSOR COMMUNICATION
- 4.1 NANOSECOND CLOCK
- 128 MWORD STATIC OR 256 OR 512 MWORD DYNAMIC COMMON MEMORY
- UP TO 40 PERIPHERAL DEVICES INCLUDING:
  - UP TO 9 DS-40 SUBSYSTEMS TOTALING 36 DD-40 DISK STORAGE UNITS (72 WITH SHADOWING)
  - UP TO 36 DD-49 DISK DRIVES
  - UP TO 8 HIGH-SPEED EXTERNAL 100 Mbyte/s CHANNEL (HSX-1) CONTROLLERS
  - UP TO 16 EXTERNAL I/O CONTROLLERS SUCH AS:
    - FRONT-END INTERFACES (FEI)
    - CRAY-2 TAPE CONTROLLERS (CTC)
    - OR HYPERchannel ADAPTORS (A130)
- MAINTENANCE CONTROL CONSOLE (MCC)

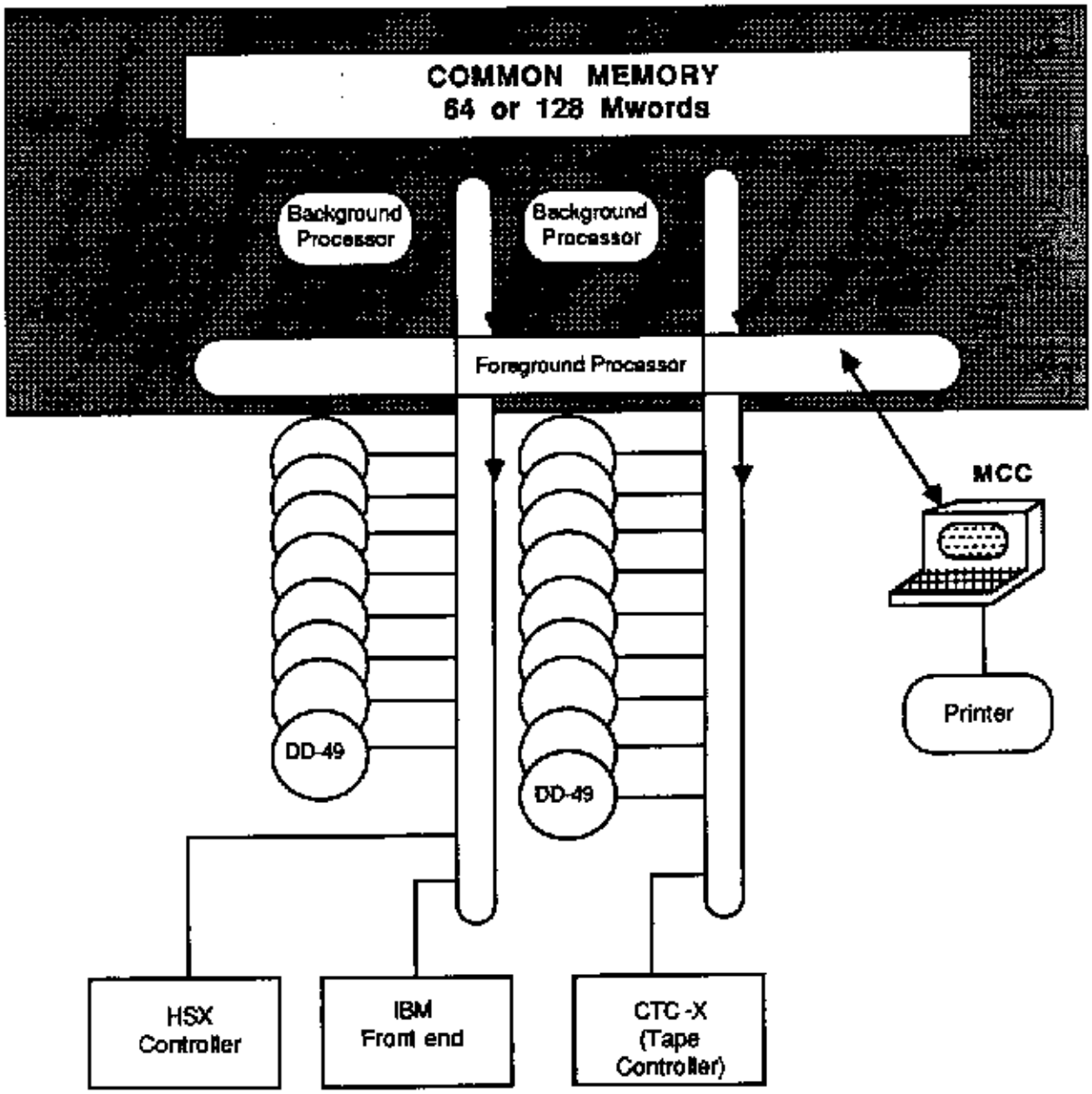
CRAY-2 FOUR - CPU SYSTEMS



CRAY-2 TWO CPU ARCHITECTURE

- TWO IDENTICAL BACKGROUND PROCESSORS
  - 16K 64-BIT WORDS LOCAL MEMORY PER PROCESSOR
  - SCALAR PROCESSING
  - VECTOR PROCESSING
  - INTERPROCESSOR COMMUNICATION
- 4.1 NANOSECOND CLOCK
- 64 OR 128 MWORD STATIC COMMON MEMORY
- UP TO 20 PERIPHERAL DEVICES INCLUDING:
  - UP TO 4 DS-40 DISK SUBSYSTEMS TOTALING 16 DD-40 DISK STORAGE UNITS (32 WITH SHADOWING)
  - UP TO 18 DD-49 DISK DRIVES
  - UP TO 4 HIGH-SPEED EXTERNAL 100 Mbyte/s CHANNEL (HSX-1) CONTROLLERS
  - UP TO 8 EXTERNAL I/O CONTROLLERS SUCH AS:
    - FRONT-END INTERFACES (FEI)
    - CRAY-2 TAPE CONTROLLERS (CTC)
    - OR HYPERchannel ADAPTORS (A130)
- MAINTENANCE CONTROL CONSOLE (MCC)

CRAY-2 TWO - CPU SYSTEMS



BACKGROUND PROCESSORS

Each background processor consists of a:

- COMPUTATION SECTION
- CONTROL SECTION
- HIGH-SPEED LOCAL MEMORY

COMPUTATION SECTION CHARACTERISTICS

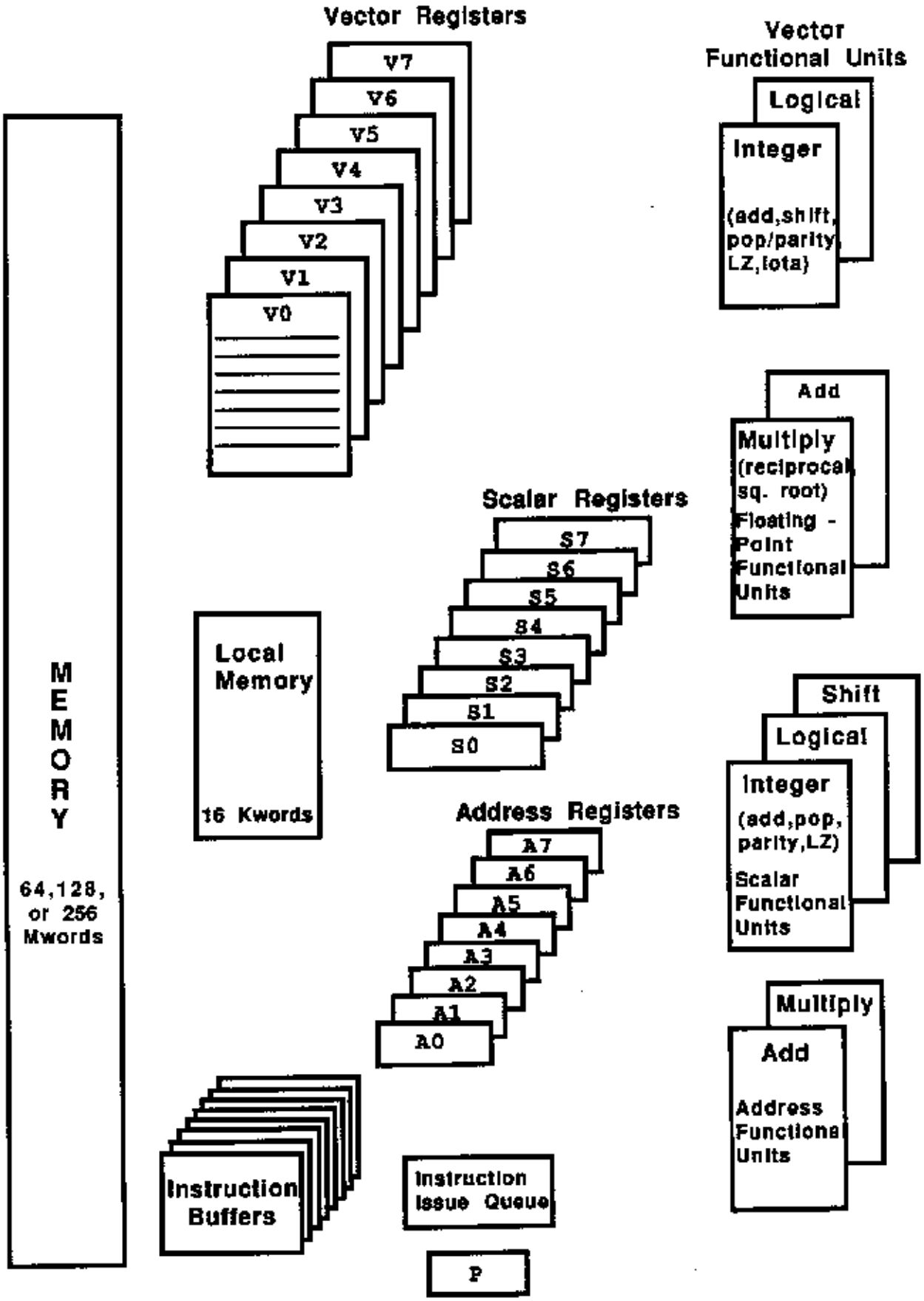
- Address and arithmetic registers
  - Eight 32-bit address (A) registers
  - Eight 64-bit scalar (S) registers
  - Eight 64-element vector (V) registers; 64 bits per element
- Address functional units
  - Add/subtract
  - Multiply
- Scalar functional units
  - Shift
  - Logical
  - Integer
    - Add/subtract
    - Population/parity
    - Leading zero count
- Vector functional units
  - Logical
  - Integer
    - Shift
    - Add/subtract
    - Population/parity
    - Leading zero count
    - Compressed iota
- Floating-point functional units
  - Add/subtract
  - Multiply/reciprocal approximation/square root
- Gather and scatter operations to and from common memory

CONTROL SECTION CHARACTERISTICS

- 8 instruction buffers (32 words each)
- 128 basic instruction codes
- 32-bit program address register
- 32-bit base address register
- 32-bit limit address register
- 32-bit status register
- 64-bit real time clock
- 8 semaphore flags for interprocessor synchronization

LOCAL MEMORY CHARACTERISTICS

- 16,384 64-bit words per CPU
- access time in clock periods:
  scalar - 4 TO 7 vector - 10 + vector length (vl)
- can overlap access with common memory or functional units
- A, S, and V register access (replaces B and T registers)



CRAY-2 Background Processor Simplified Block Diagram

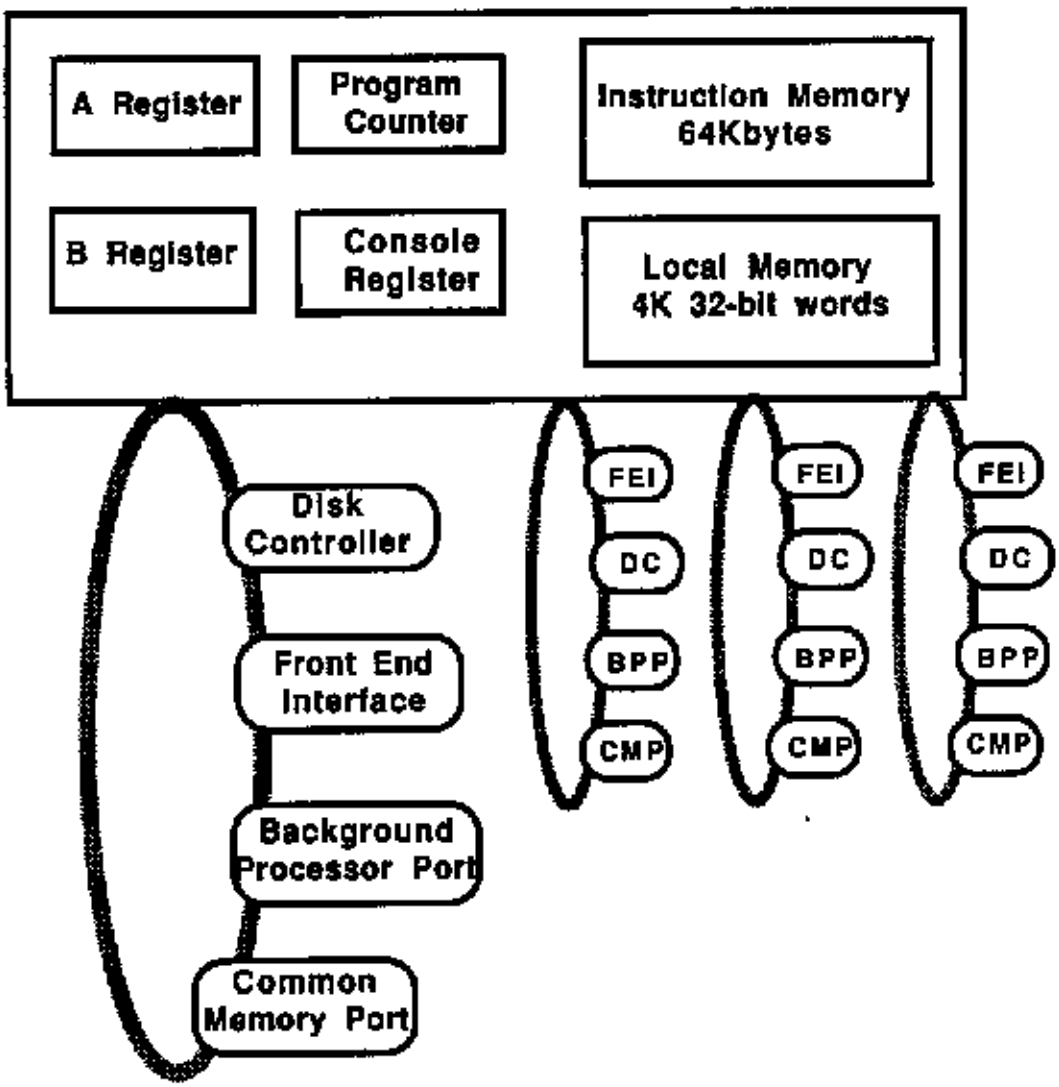
FOREGROUND PROCESSOR

- INSTRUCTION MEMORY
- LOCAL DATA MEMORY
- ARITHMETIC FUNCTIONS
- REAL-TIME CLOCK
- ERROR CHECKING
- INSTRUCTION ISSUE MECHANISM
- INSTRUCTION SET

FOREGROUND PROCESSOR CHARACTERISTICS

- OVERALL SYSTEM SUPERVISION
  - BACKGROUND PROCESSORS
  - I/O CONTROLLERS
- 32-BIT PROCESSOR
  - A AND B REGISTERS
  - LOCAL DATA MEMORY 4096 WORDS
  - INSTRUCTION MEMORY 64K BYTES
  - INTEGER ARITHMETIC
- PERIPHERAL CONTROLLER INTERFACES
  - DC-3: DD-40 Disk Controller
  - DC-2: DD-49 Disk Controller
  - FE-3: CONNECTION TO FEI, CTC, OR A130
  - HSX-1: HIGH-SPEED EXTERNAL CHANNEL
- FOUR COMMUNICATION CHANNELS
  - (TWO ON 2-CPU MODELS)
  - 500 Mbyte/s EACH
  - CAPACITY OF 36 DD-49 DRIVES (43.2 GBYTES)
  - LINK FOREGROUND AND BACKGROUND PROCESSORS AND CONTROLLERS
- SYSTEM DEADSTART

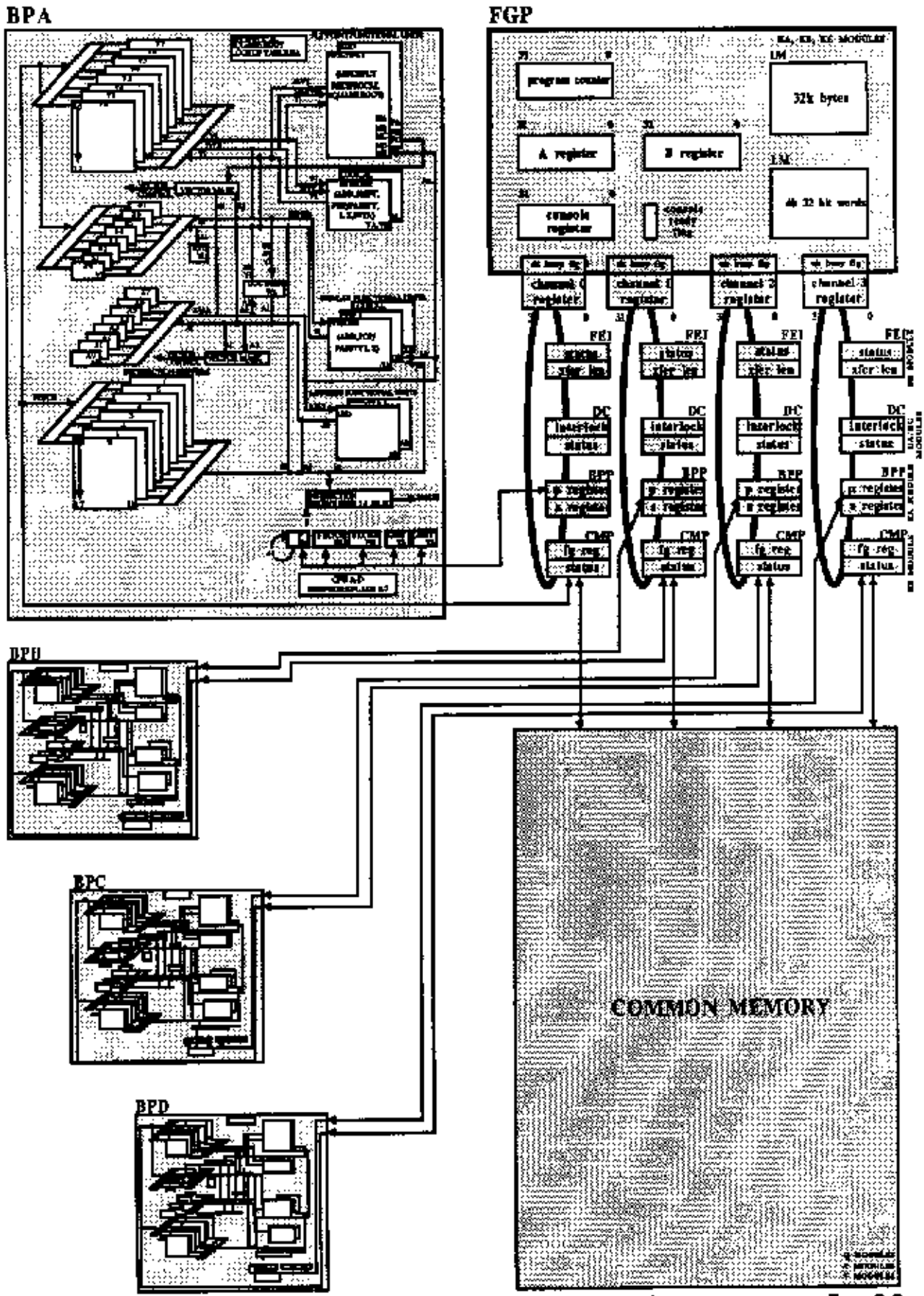
CRAY-2 Foreground Processor / Channel Loop Architecture Simplified Block Diagram



FOREGROUND SYSTEM

- PROVIDES COMMUNICATION BETWEEN:
  - BACKGROUND PROCESSORS
  - FOREGROUND PROCESSORS
  - DISK CONTROLLERS
  - FRONT-END INTERFACES
- FOREGROUND CHANNEL PORTS:
  - CONTROL DATA TRANSFERS BETWEEN
    - COMMON MEMORY AND PERIPHERAL EQUIPMENT
  - MONITOR AND CONTROL BACKGROUND
    - PROCESSORS
- PROVIDES ACCESS AND CONTROL FOR UP TO 36 DISK STORAGE UNITS (UP TO 18 ON TWO-CPU MODELS)
- SUPERVISES OVERALL SYSTEM ACTIVITY
- USES MAINTENANCE CONTROL CONSOLE TO:
  - DEADSTART
  - MONITOR SYSTEM OPERATION

CRAY-2 SYSTEM



DS-40 DISK SUBSYSTEM

- 5200 MBYTES PER DRIVE
- UP TO 36 DISK STORAGE UNITS
  = 187.2 Gbytes (9 DS-40 DISK SUBSYSTEMS)
  (374.4 Gbytes WITH DAISY CHAINING)
- 20 MBYTES/S BURST RATE
- AVERAGE ACCESS TIME 16 ms
- MEDIA DEFECTS HANDLED BY "SKIP DEFECT"
- DD-40 DISK STORAGE UNIT SPECIFICATIONS
  - 208 VAC, 60HZ, 20A
  - 1150 LBS (522 KG)
  - FORCED-AIR COOLING

DD-49 DISK DRIVE

- 1200 MBYTES PER DRIVE
- UP TO 36 DRIVES = 43.2 Gbytes
- 12 MBYTES/S BURST RATE
- AVERAGE ACCESS TIME 16 ms
- WINCHESTER TYPE HDA PLATED MEDIA
- TWO SPARE SECTORS PER TRACK
- 208 VAC, 60HZ, 20A
- 844 LBS (383 KG)
- FORCED-AIR COOLING

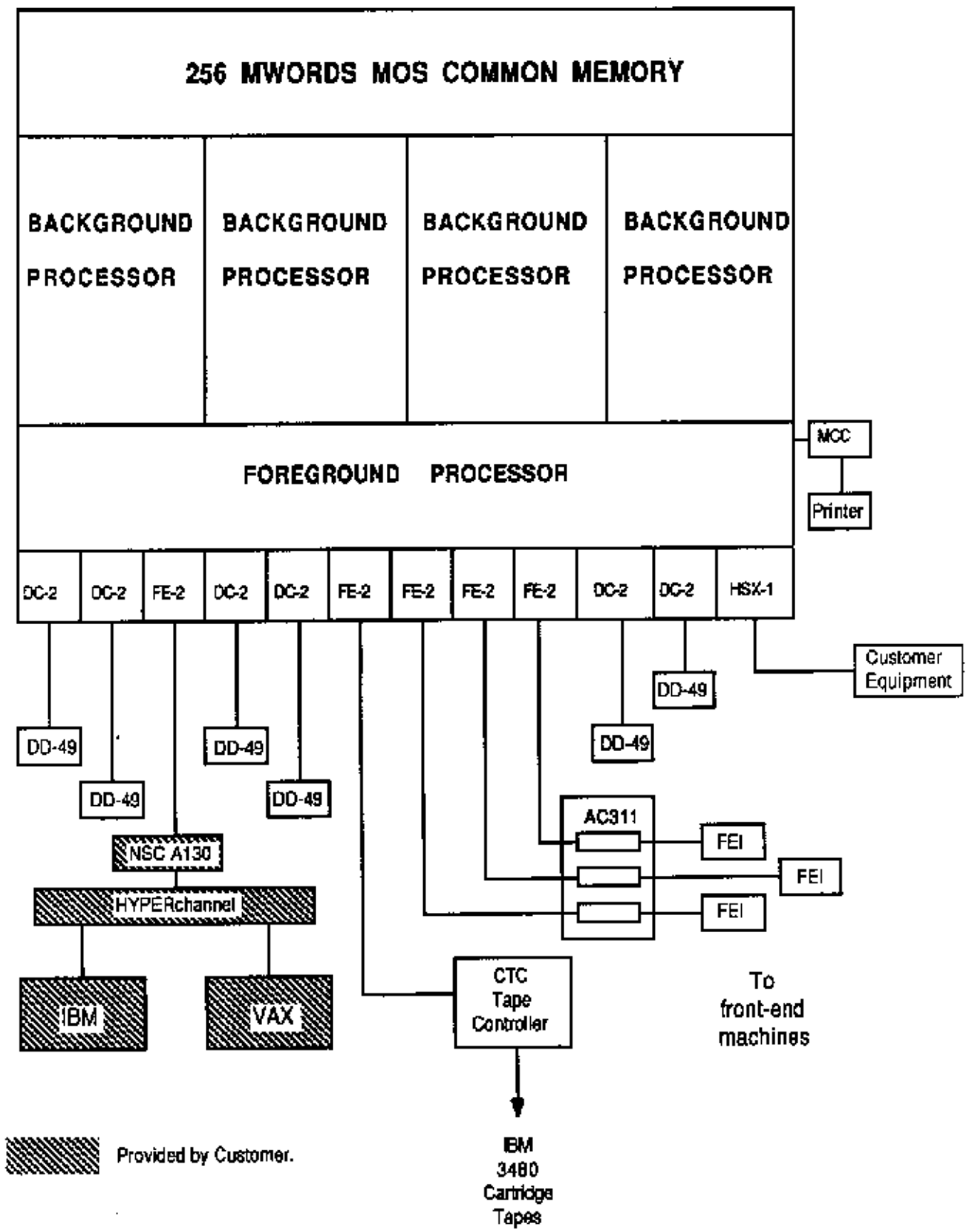
HSX HIGH-SPEED EXTERNAL CHANNEL

- CAN CONNECT TO CUSTOMER-FURNISHED GRAPHICS AND SIMULATION DEVICES
- ALLOWS CONNECTING TO OTHER CRAY-2s OR A CRAY X-MP SYSTEM
- TRANSFER RATE OF UP TO 100 Mbytes/s
- CHANNEL PROTOCOL IS PEER TO PEER (NOT MASTER-SLAVE)

FRONT-END CHANNEL

- ALLOWS CONNECTION TO A VARIETY OF FRONT-END COMPUTER SYSTEMS
- UP TO 16 FRONT-END SYSTEMS CAN BE ACCOMMODATED IN A POINT-TO-POINT CONFIGURATION
- AN UNLIMITED NUMBER OF SYSTEMS CAN BE CONNECTED USING COMMERCIALLY AVAILABLE BUS NETWORKS (e.g., HYPERchannel)

CRAY-2
SAMPLE CONFIGURATION



CRAY-2 TAPES CONTROLLERS

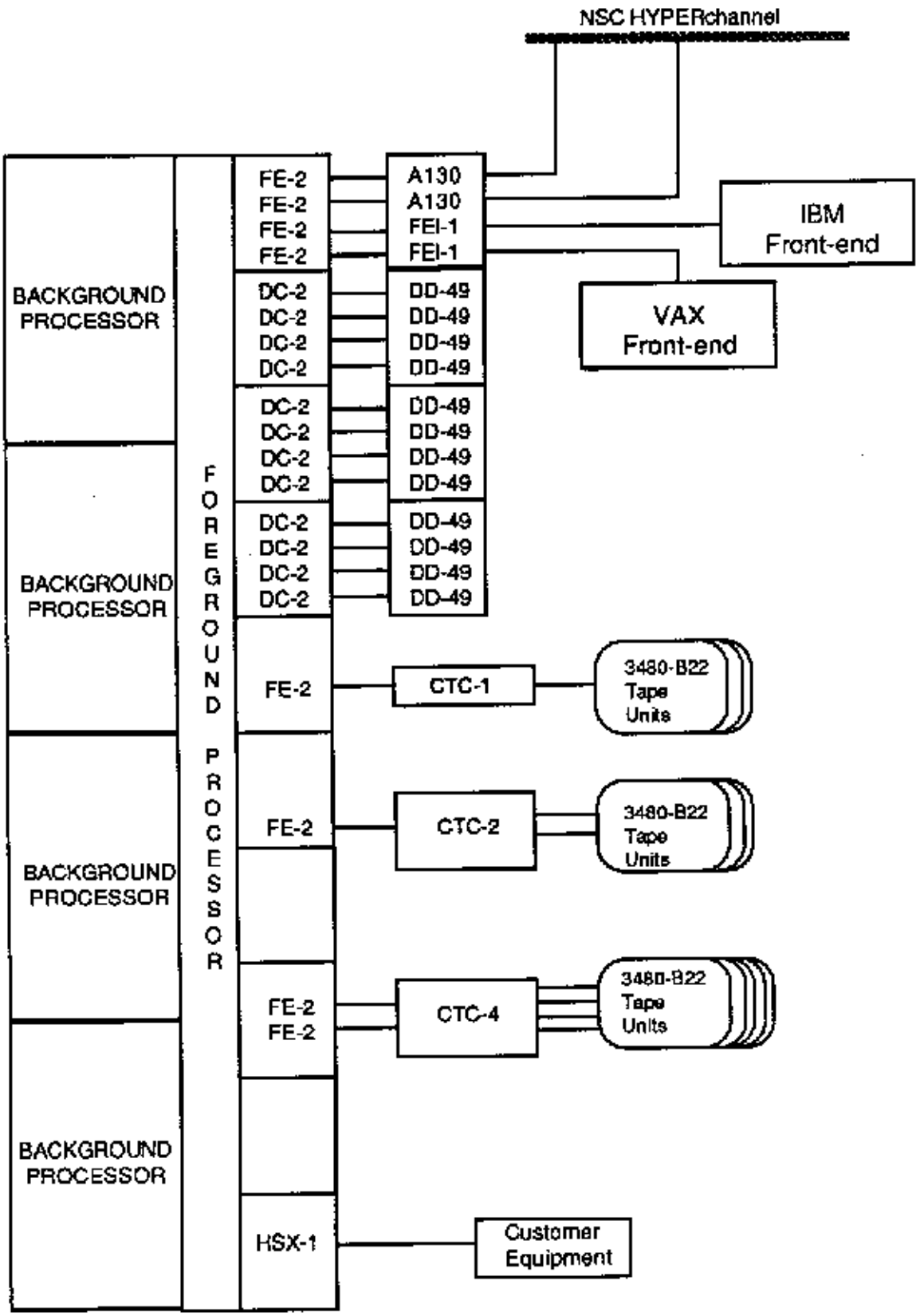
- ENABLES A CRAY-2 SYSTEM TO INTERCONNECT WITH ONE OR MORE IBM 3480 MAGNETIC TAPE CARTRIDGE SUBSYSTEMS
- SYSTEM USES A VMEbus ARCHITECTURE AND 68000 MPU

CONTROLLER	FE-2	DATA STREAMING CHANNELS
CTC-1	1	1
CTC-2	1	2
CTC-4	2	4

CRAY-2 TAPE PERFORMANCE

- CTC-1 WITH ONE STREAM (DRIVE):
  2.5 Mbytes/s
- CTC-2 WITH TWO STREAMS (DRIVES):
  5.0 Mbytes/s
- CTC-4 WITH FOUR STREAMS (DRIVES):
  10.0 Mbytes/s

Sample CRAY-2 Tape Configurations



CRAY-2 PURCHASE PRICES
(Made available by instructor)

SECTION 6

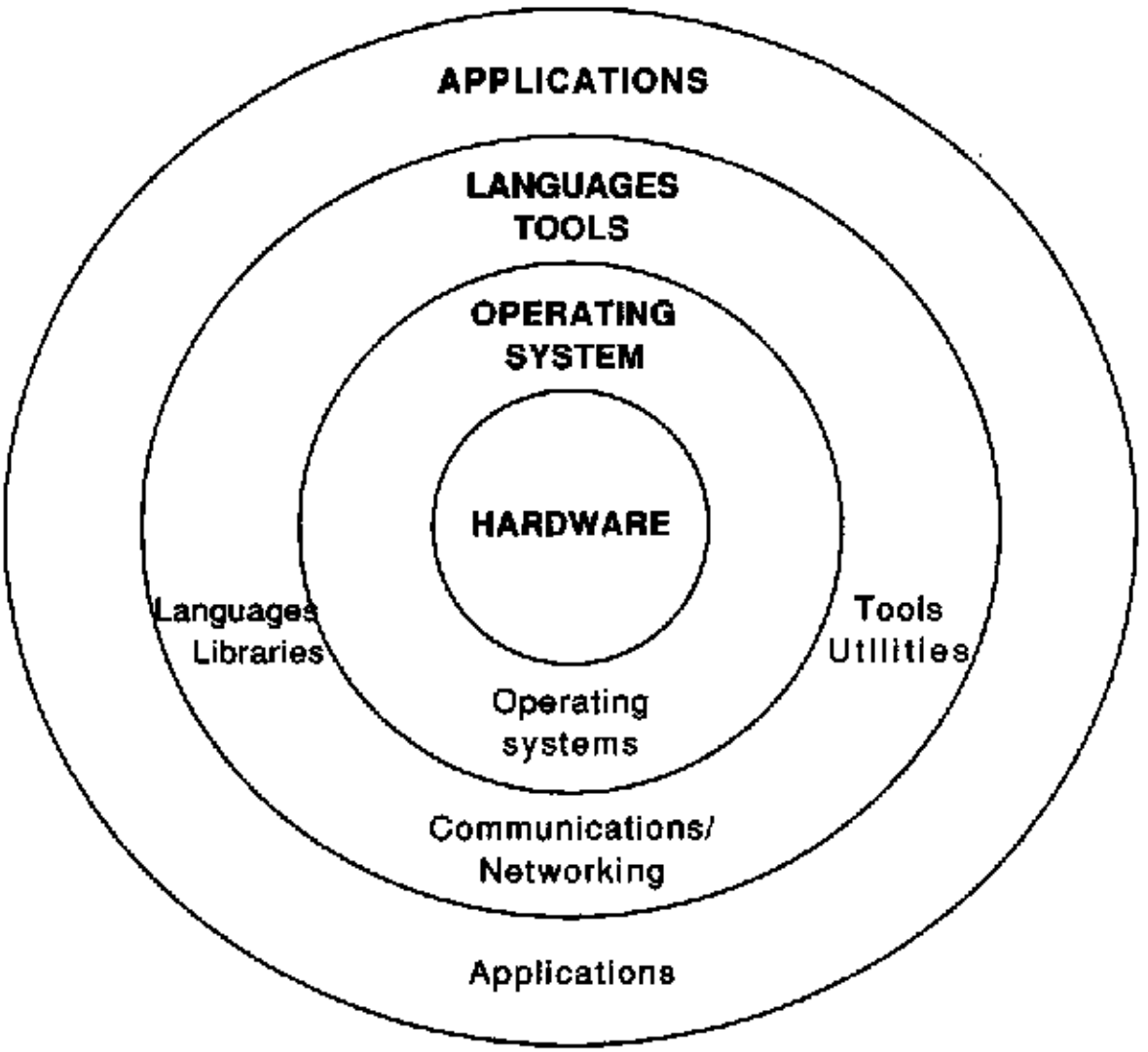
Cray Product Familiarization

THE COS OPERATING SYSTEM

Cray Research, Inc. Software Training

THIS PAGE IS INTENTIONALLY LEFT BLANK

SOFTWARE STRUCTURE



Most computer **software** can be placed within categories or layers as the drawing above shows. Many of the same **application programs** such as word processors or even car crash simulation programs can work on different computers as long as the software on the inner layers is able to convert what the program is asking the computer to do into something the hardware can understand. This is **system software** which includes tools, utilities, languages, libraries, communications/networking, and operating system code necessary to get the job done.

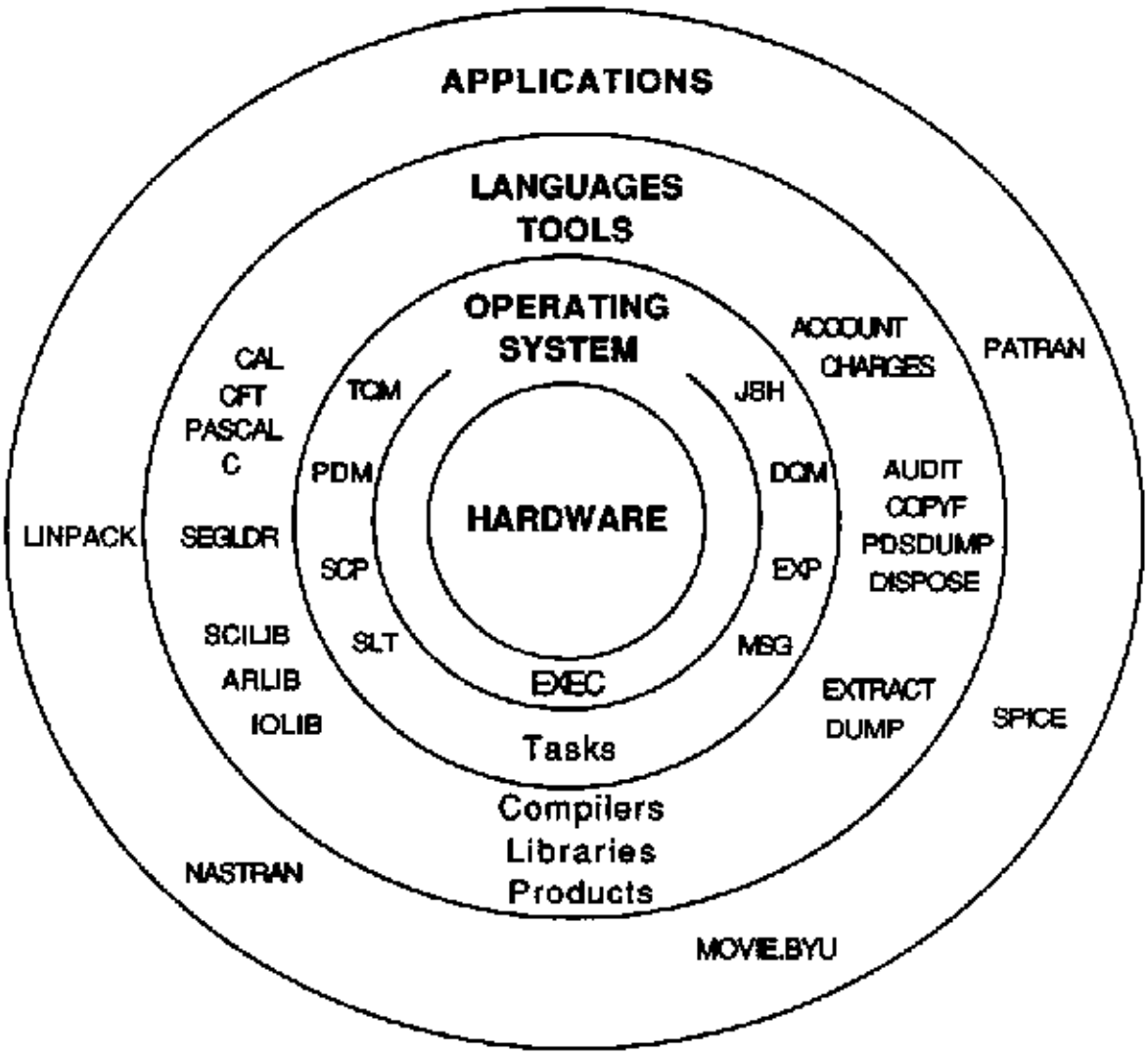
Similar drawings to this one will be used to show some of the complementary parts of the COS and UNICOS operating systems.

CRAY OPERATING SYSTEM (COS)

- Efficient, high-throughput batch-oriented system
- Works in conjunction with the IOS Operating System on mainframes with an IOS
- Structured as a monitor-mode executive and a set of user-mode system tasks
- Common to CRAY-1 and CRAY X-MP
- Simple English-like Job Control Language (JCL)
- Permanent Dataset Management
- Dynamic disk allocation/deallocation
- On-line tape support
- Flexible job class scheduler
- Information logging at job and system levels
- Installation-controlled accounting parameters
- Common interface to front-ends and system operator
- Job recovery following system interruption
- Interactive capability
- Automatic dataset archiving
- Multiprogramming/multiprocessing/multitasking
- SuperLink/ISP
- Many levels of security

COS development efforts will end soon because of emphasis on the UNICOS Operating System.

COS STRUCTURE



COS COMPONENTS

COS is run by a central program called EXEC which aligns each incoming job with the resources it needs. It also makes sure that jobs don't interfere with each other and that each one gets an opportunity to run.

Another background program jobs need frequently is Control Statement Processor (CSP) which deciphers Job Control Statements (JCL) in jobs.

To help EXEC out, a group of system tasks collectively known as System Task Processor (STP), provide the routine chores jobs regularly need, such as disk I/O, job scheduling, and many more. One of the tasks, the Exchange Processor (EXP), acts as a messenger between a job and EXEC. This prevents jobs from directly affecting EXEC.

The other tasks include:

Startup - used only to restart the computer

Station Call Processor (SCP) - Handles protocol for communication with front-ends

Permanent Dataset Manager (PDM) - Maintains the catalog of datasets stored on disk

Disk Error Correction (DEC) - Handles error recovery for disks

Disk Queue Manager (DQM) - Handles disk I/O requests and disk space management

Message Manager (MSG) - Formats log messages and initiates I/O to system and user logs

Message Processor (MEP) - Initiates memory error log messages

System Performance Monitor (SPM) - Collects and formats system performance stats

Job Scheduler (JSH) - Allocates memory and CPU resources to user jobs and tasks

Job Class Manager (JCM) - Assigns jobs to classes according to the job class structure

Tape Queue Manager (TQM) - Handles tape I/O requests

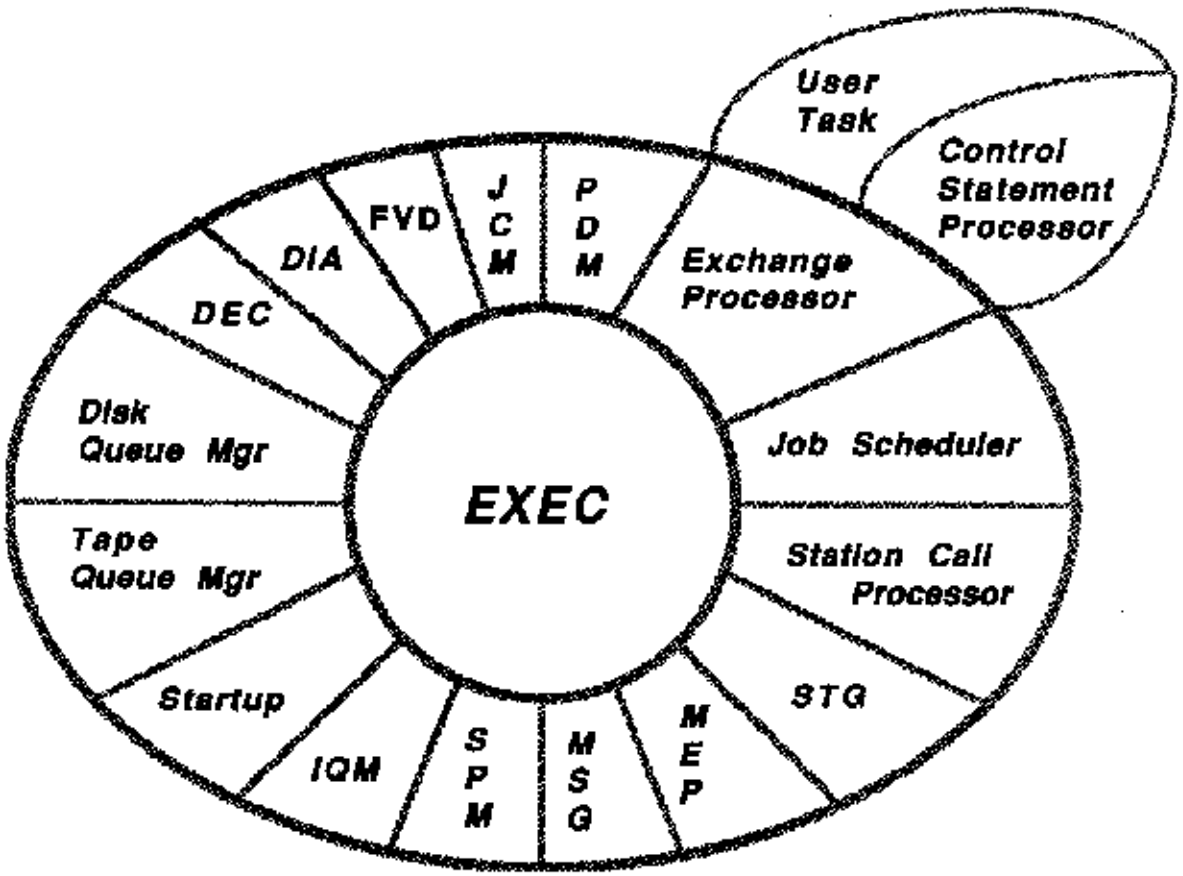
Stager (STG) - Works with SCP to handle the movement of data to and from front ends

Flush Volatile Device (FVD) - On request, moves data from volatile to non-volatile mas storage devices (e.g. from SSD to disk)

Superlink Task (SLT) - Handles I/O requests for Superlink/ISP

Diagnostic (DIA) - Initiates the Disk Online Utility which allows disk maintenance while the operating system continues to run

COS Components



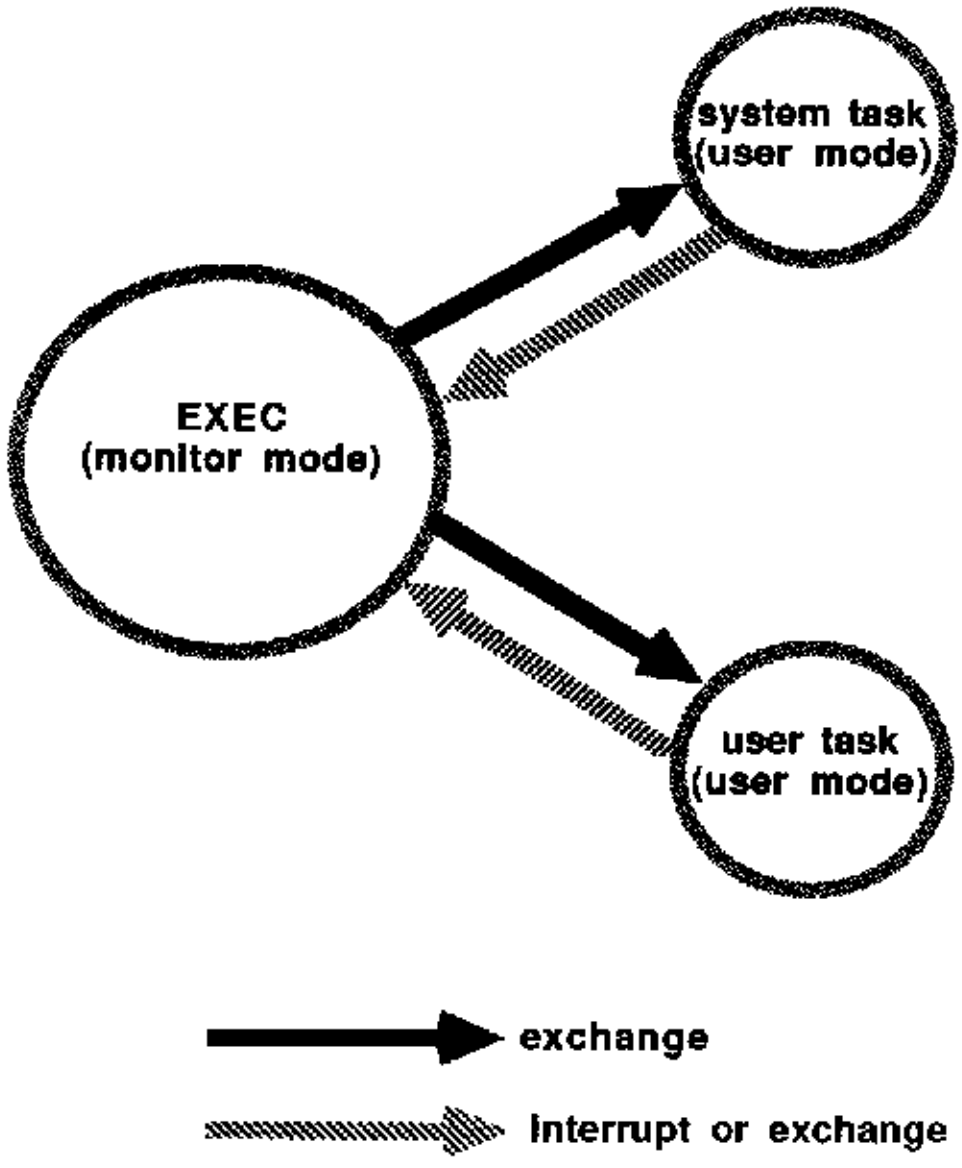
COS EXECUTION

COS uses a swapping technique in which EXEC gets switched to on every request. For example, if a job needs some data from disk, EXEC gets notified of this and asks the appropriate tasks (i.e., PDM and DQM) to do the work. They will notify EXEC when they are finished.

This method of exchanging back and forth allows EXEC to initiate tasks for jobs, then search for other jobs needing services. This is especially important since COS is **single-threaded**. This means that EXEC can only be connected to one job at a time to prevent confusion. This becomes even more important in multi-CPU machines.

COS is written in Cray Assembler Language (CAL).

COS execution



THIS PAGE IS INTENTIONALLY LEFT BLANK

SECTION 7

Cray Product Familiarization

The UNICOS Operating System

Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK

WHAT IS UNICOS?

The UNICOS Operating System is the newest operating system at Cray Research, Inc. It has the following characteristics:

- Based on UNIX System V from AT&T
- Enhanced to support Cray supercomputers
- Runs on Cray-1(with IOS), Cray X-MP, Cray-2, and CRAY Y-MP systems
- Will run on future Cray systems

The goals of UNICOS development are to:

- Deliver supercomputer performance
- Provide full functionality (full use of the hardware features)
- Follow System V enhancements
- Provide common products with the Cray Operating System (COS) for migration purposes

WHY IS IT BASED ON UNIX?

There are many reasons why the UNIX operation system was chosen as a base for UNICOS. They include:

- UNIX is fast becoming an industry standard
- Speed of implementation
- Portablility (written mostly In C language)
- Functionality
- Performance
- Existing applications and utilities (no need to rewrite them)

CRAY UNICOS COMPATIBILITY

The goals of UNICOS include:

- Maintain consistent interface to existing and new machines
- Preserve customer software investment

Cray Research's commitment to UNICOS compatibility across all hardware ". . . is to have the same interface for user calls, system calls, library calls, and languages. Exceptions will occur due to architectural differences or to performance needs which are dependent on architectural differences."

HISTORY OF UNIX

AT&T System V

The UNIX operating system was first developed in 1969 at the Bell Laboratories in New Jersey. Many people have contributed to its development leading to the most recent release, System V.

UNIX was developed to provide a time-sharing system which allowed more than one person to use the computer at the same time (multi-user) and it allowed a person to communicate directly with the computer via a terminal (interactively).

UNIX designers took advantage of other multi-user operating systems in developing UNIX. Researchers developed UNIX to allow a group of people working together on a project to share data and programs.

It has become increasingly expensive for hardware manufacturers to develop and support proprietary operating systems as the cost of hardware continues to drop. Likewise, application software manufacturers can't afford to convert their products to run on many different proprietary operating systems. UNIX fills the needs of both. It can fairly easily be adapted to different computer hardware and software programs can be written to the operating system, not the specific machine. Therefore, a program which runs on one UNIX machine can easily run on another UNIX machine.

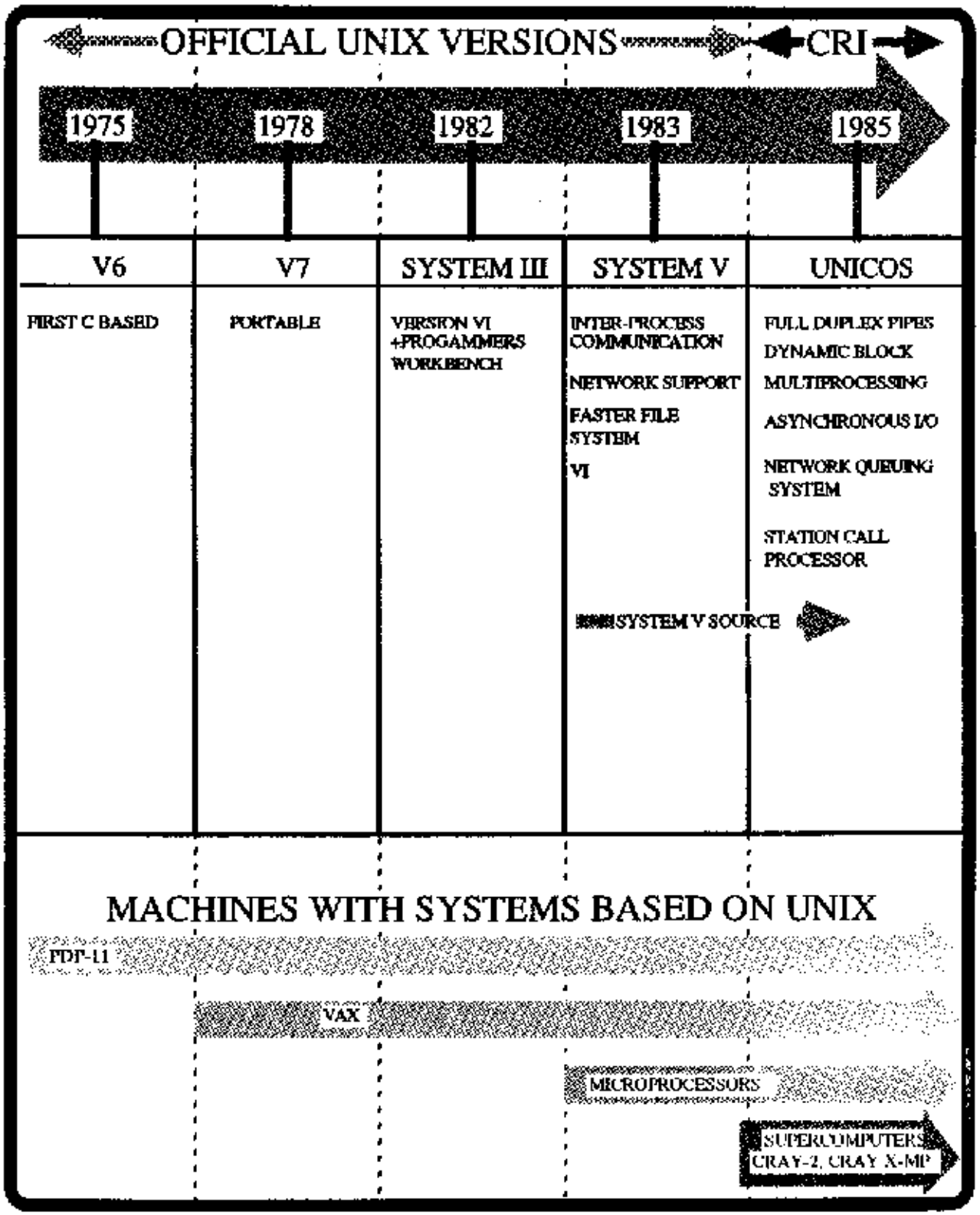
Another benefit of UNIX is that it is written in C, a high-level, machine-independent programming language.

The Berkeley Influence

When UNIX became widely available in 1975, Bell Labs offered it to educational institutions at a minimal cost. The schools used it in their computer science programs and, thus, it became popular. Graduating students then brought their UNIX experience to the commercial world.

The Computer Science Department of the University of California at Berkeley made significant additions and changes to UNIX. In fact, they made so many popular changes that one version is called the Berkeley Software Distribution (BSD) of the UNIX system. System V from AT&T has adopted many of the features developed at Berkeley.

UNICOS EVOLUTION



UNICOS Features and Functions

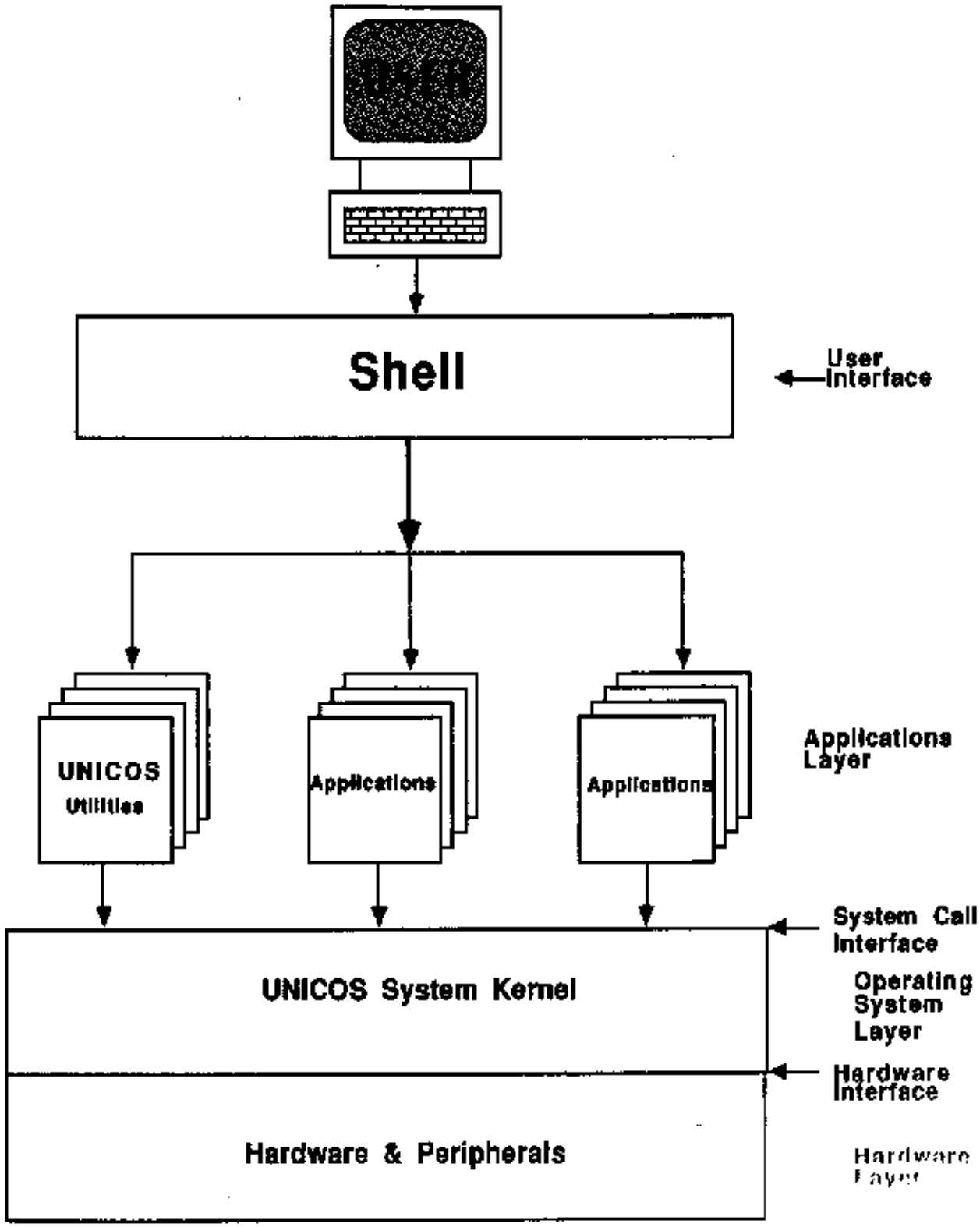
User level:

- Shell
- File/Directory System
- Commands/Utilities/Compilers
- Pipes/Filters
- I/O Redirection
- Shell Programs/Scripts
- Applications

System Level:

- Resource Management
- Process Management
- File Management
- I/O Management

THE STRUCTURE OF UNICOS



The KERNEL

- Memory resident part of the operating system
- Operating system functions are accessed by utilities and applications through system calls
- A processing system
  - Controls CPU scheduling
  - Memory management
  - Process management

The UNICOS Shell and Shell Scripts

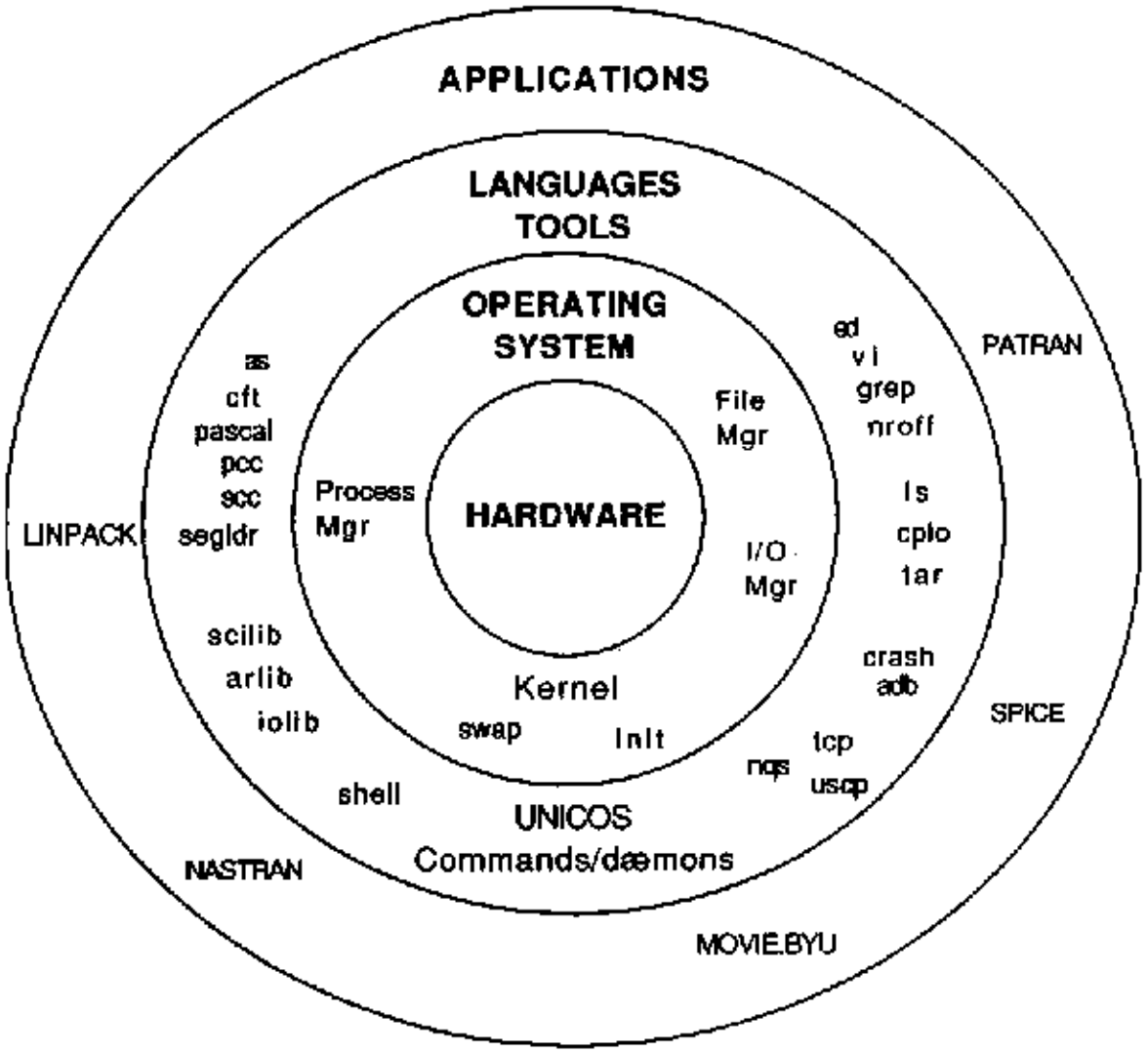
The Shell is a command interpreter which also has many features of a high-level programming language. The shell's job is to connect the user with the requested program or utility.

UNIX (and UNICOS) is powerful in that it allows combining several commands (utilities) in a single command line. **A Shell program (or script)** is a file which contains one or many of these command lines which can be invoked by simply giving the filename as if it were a single command. Within the shell script, features similar to a high-level language allow user-supplied parameters and conditional statements (e.g., if, then).

From a single terminal, many shells (called layers) can be controlled and run in the background.

The benefit of all this is that, by writing appropriate shell scripts, the appearance of UNIX to the user can be tailored for special groups or purposes. For example, a shell script can make UNIX appear to be a simplified menu-driven system for data entry to be used by clerical personnel. In this case, the user does not need to know any UNIX commands.

UNICOS STRUCTURE



UNICOS FEATURES

The File System

- Hierarchical
  All disk space is grouped into manageble file system pieces that become branches of the file system

- Access validation
  Provides a file protection scheme

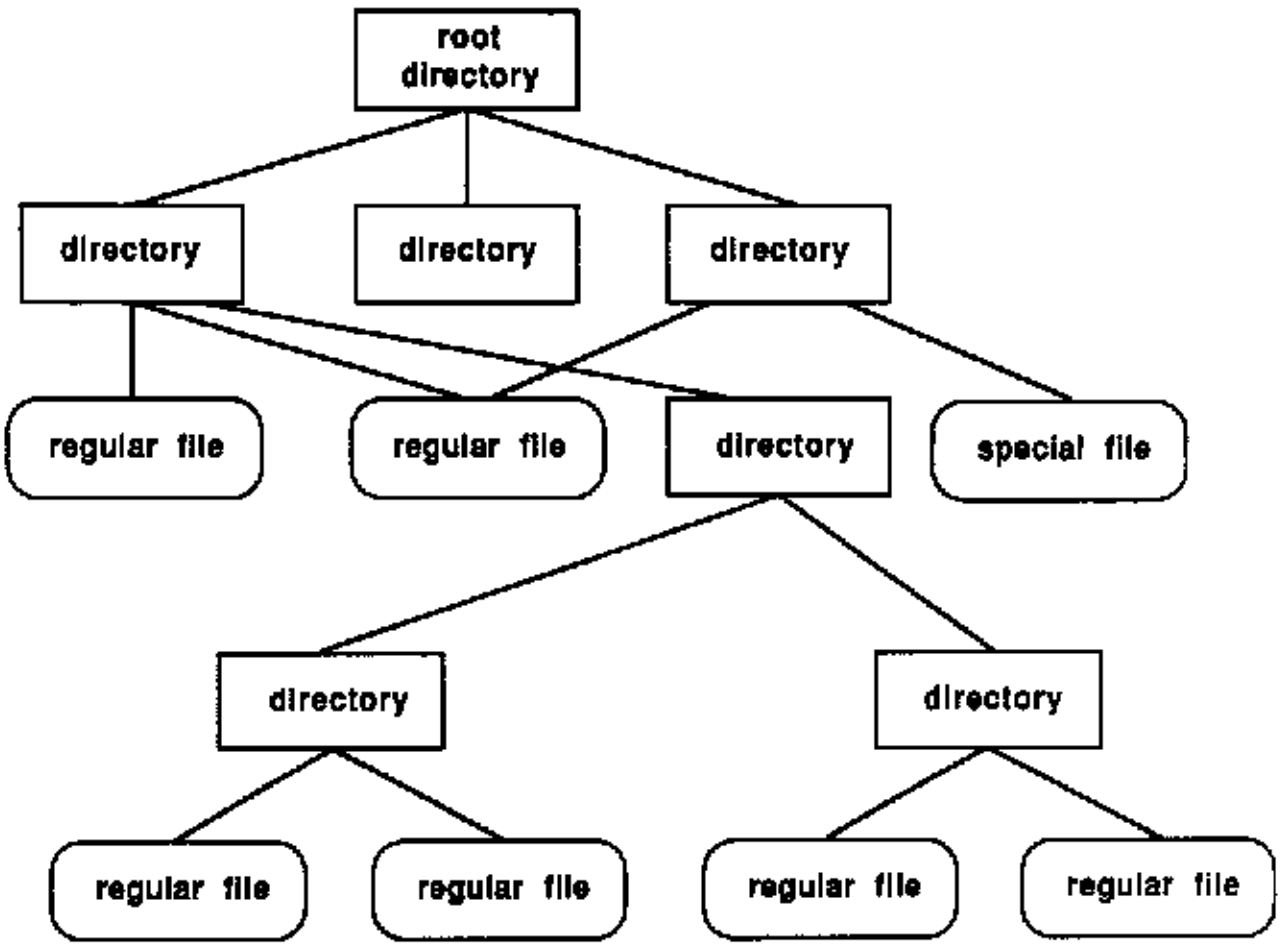
- Accessing I/O hardware is like accessing an ordinary disk file (device independent I/O)

- File types:
  Regular files (for user programs, text, data, or binary files)

  Directory files (for grouping regular files together)

  Special files (i.e., device drivers such as terminals, disks, tape, or printers)

HIERARCHICAL FILE STRUCTURE



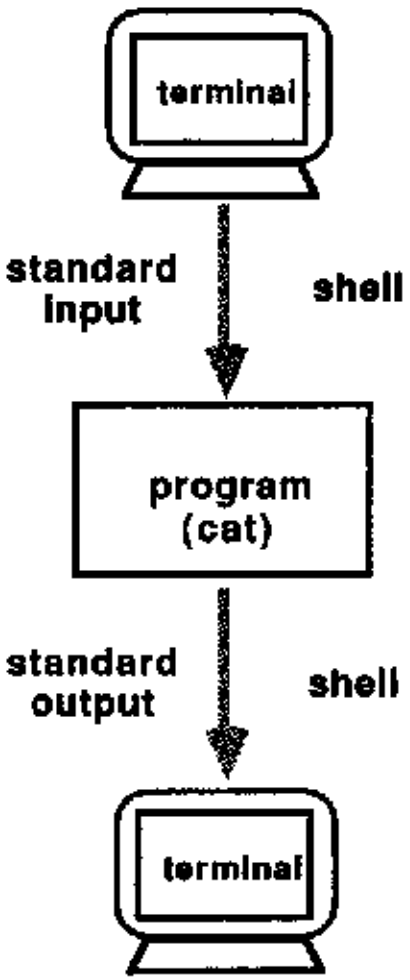
USER FEATURES

I/O Redirection

Without special instructions, the shell takes its **standard Input** from the terminal keyboard (that is, from the user who types it in) and sends its **standard output** to the terminal screen. In the example on the next page *$* is the shell prompt, *cat* is the name of a utility which lists (con**cat**enates) the contents of a file to the standard output (the screen). By typing in *cat* alone the program waits for the user to type text on the keyboard. Everything being typed is shown on the screen. Then, whenever the user presses the Return (or Enter) key, the line of information just typed is "echoed" to the screen.

This is not the usual use of the *cat* utility, but is shown to help explain what standard input and standard output are. More appropriate uses for the *cat* utility are on the following pages.

Standard I/O



example:

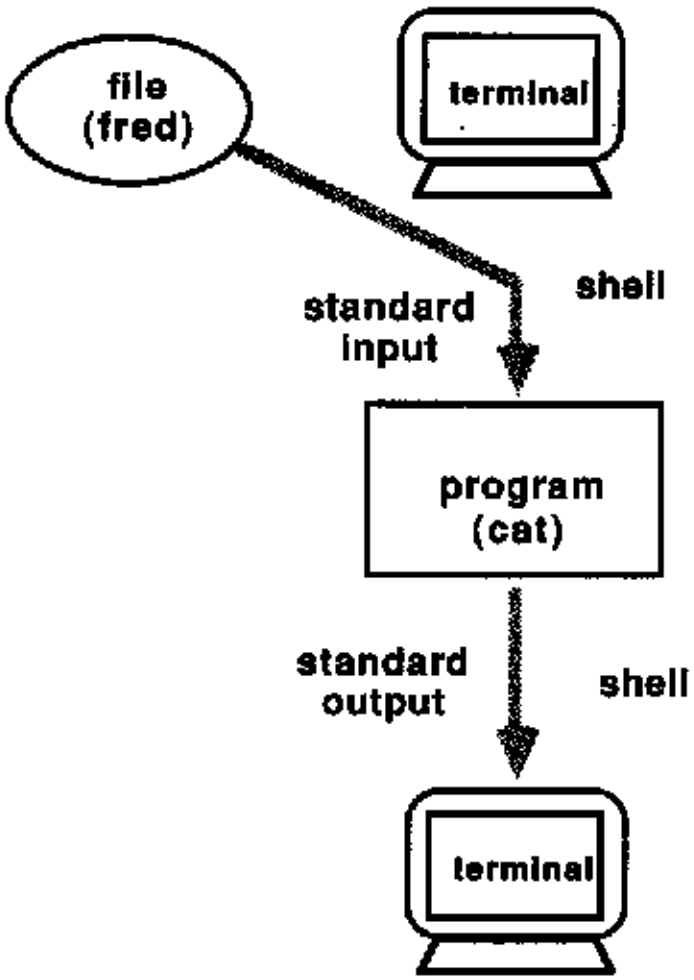
$ cat

USER FEATURES

Input Redirection

In this example, the *redirect input* symbol (<) is used to send the contents of the file *fred* as input to the *cat* utility. If the file *fred* (the redirected input) contains any data, it will be displayed on the screen (standard ouput).

Input Redirection



example:

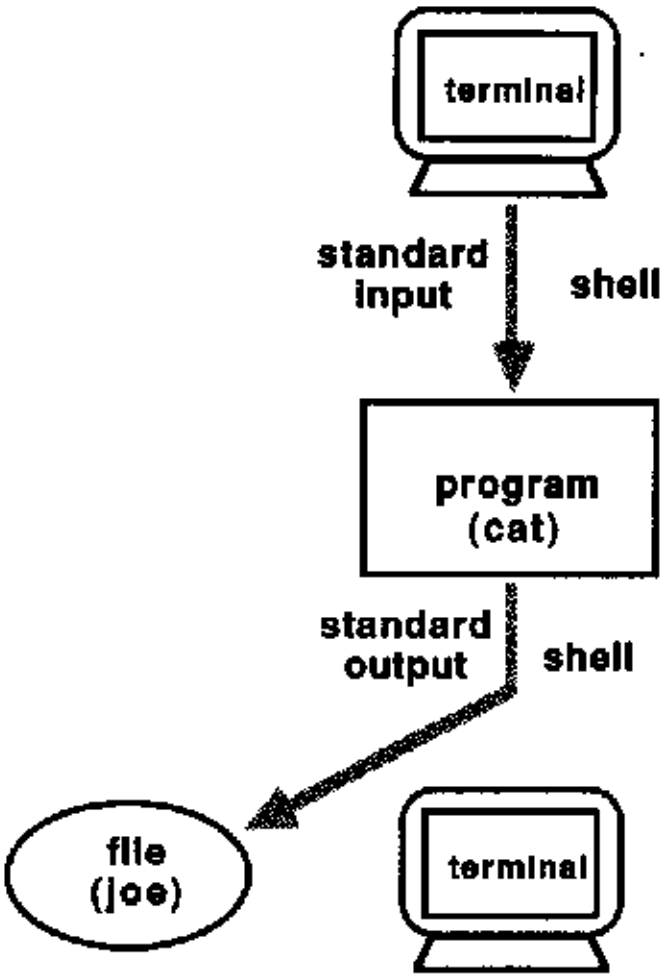
$ cat < fred

USER FEATURES

Output Redirection

In this example, the *redirect output* symbol (>) is used to send the output from the *cat* utility to the file *joe* instead of the standard output (the screen). The input to the *cat* utililty will still be the standard input; that is, whatever is typed on the keyboard will be sent by *cat* into the file *joe*.

Output Redirection



example:

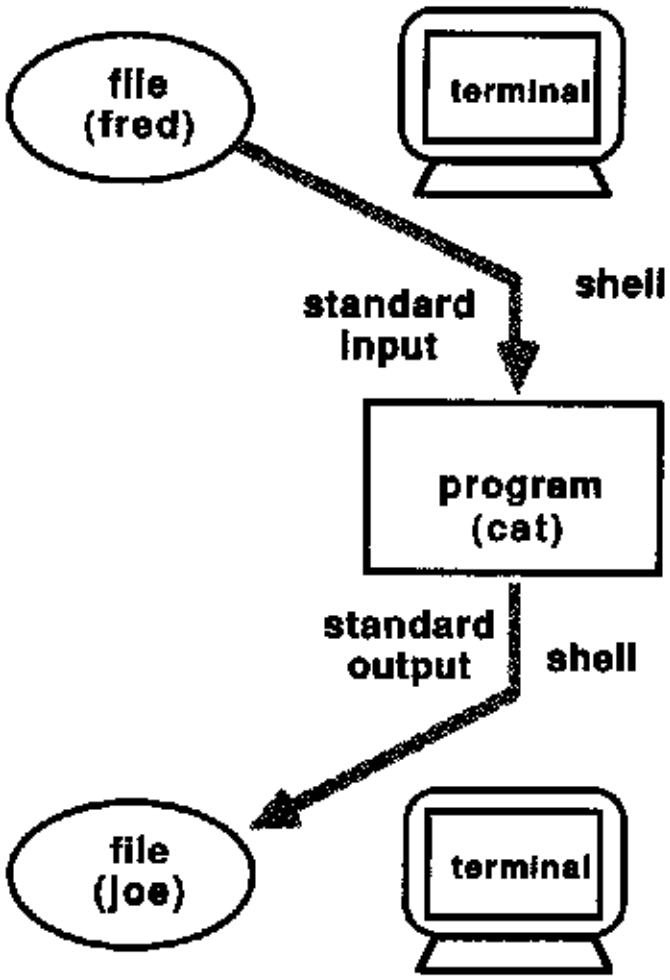
$ cat > joe

USER FEATURES

I/O Redirection

In this example, the contents of file *fred* will be written into the file *joe* by the *cat* utility. Essentially, *fred* is the Input to *cat* and *joe* is the output from *cat*. No input from the keyboard (standard input) will be taken by *cat*, nor will *cat* send any output to the screen (standard output).

Input/Output Redirection



example:

$ cat < fred >joe

PIPES

In the world of UNIX, a utility which reads standard input, does something to that input, and writes it to standard output is called a **filter**. Stringing several of these utilities (filters) together in one command line would be more efficient for both the user and the operating system.

This is possible in UNIX (UNICOS) because it can use **pipes**. Pipes simply take the output from one program or utility and send it as input to another program or utility. This is done on a single command line using the pipe symbol (|) between commands.

In the example on the next page, *grep* is a utility which searches for and displays lines in a file which contain a certain pattern (called a *string*). The *-i* symbol is called an option, which modifies the way in which *grep* works. A utility can have many different options and it is possible to specify several of them at once. This particular option specifies to *grep* to ignore case; that is, to match the given pattern with either upper or lower case letters in any combination.

The next thing in the command line, *joe*, is the pattern to be matched. Following that is the filename of the file to be searched,*/etc/passwd*. Next, the output from the *grep* operation is piped into the *sort* utility which puts all lines found into alphabetical order.

A synopsis of the whole process is as follows: the user types in the command line and *grep* looks in the file of passwords for all lines having the form joe, Joe, JOE, joebob, Billyjoe, joe007, etc. It then passes these lines to the *sort* utility which puts them in alphabetical order and sends them to the screen. The output could have been redirected to a file instead of the screen (e.g., *grep -i joe /etc/passwd|sort>**file***). This sequence might be used by the UNICOS system administrator to search through the list of all user passwords in a convenient way.

Notice that each utility by itself is general-purpose in nature. However, by stringing the utilities and their operands together with I/O redirection and pipes, it is relatively convenient to accomplish many special-purpose operations and better utilize the system.

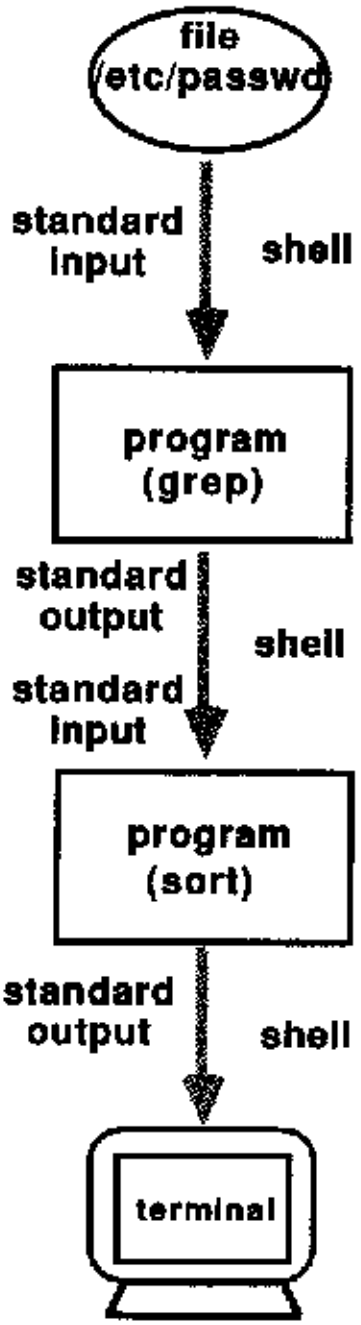
Piping is a form of multitasking built into the UNIX operating system.

Another form of multitasking in UNIX (UNICOS) is done by running programs (utilities) in the **background**. This is simply done by typing an *&* after the command. The system will go off and work on the command, but gives the user the prompt back to enter more commands.

Pipes

example:

grep -i joe /etc/passwd | sort



Enhancements to UNIX for Use on Cray Supercomputers

Since standard UNIX is designed for use on micro- and minicomputers, UNICOS required new or additonal code to take advantage of the speed and power of Cray supercomputers. Some of the areas of enhancement are defined on the next several pages.

UNICOS I/O Enhancements

Standard UNIX deals with character I/O; Cray computers typically deal with large blocks (disk sectors and tracks) at a time. Faster data retrieval is required on Cray computers so different methods of preloading data (e.g., into the SSD) are needed so the system doesn't need to wait for data.

- **Asynchronous I/O**
  Used to overlap I/O processing; an I/O request can proceed immediately while the main process simultaneously continues to execute (used for multitasking)
- **List I/O**
  - A linked list of I/O requests can be initiated with either synchronous or asynchronous control
  - used in multitasking in which an I/O list is sent to the Cray and the program continues with other work
- **Raw I/O**
  - Data goes directly to user space, bypassing the system buffers first (provides less overhead, so faster I/O)
- **Streaming**
  - Reads and writes to disk can be a continuous flow without stopping at each sector boundary; this allows data to flow at the full speed of the disk drive
- **Disk Striping**
  - Files can be spread over two or three drives (stripe group) which allows I/O speedup of that factor (up to 15 with UNICOS 5.0) since the read or write can be happening in parallel
- **SSD Support**
  - UNICOS file systems can be put on it for fast access (e.g., /tmp)
  - system buffers can be put on it (SSD cache) so data can be preloaded and ready for the program that needs it without waiting for disk I/O (when the system cache buffers are full extra data can be automatically put on the SSD instead of remaining on disk)
  - the program can directly read and write to and from the SSD without UNICOS intervention ("hot path")
- **BMR support (IOS Buffer Memory)**
  - Allows putting file system on it (for fast access, as on SSD)

UNICOS I/O Enhancements

- **Asynchronous I/O**
- **List I/O**
- **Raw I/O**
- **Streaming**
- **Disk Striping**
- **SSD Support**
  - **BMR support (IOS Buffer Memory)**

UNICOS File System Enhancements

- **Improved allocation**
  - **Multiple physical blocks per logical block**
    Standard UNIX deals with one-sector blocks; under UNICOS a number of sectors can be defined as a logical block

- **Clustering of disk partitions**
  - **Disks can be divided up into separate logical devices**
    Slice: contiguous cylinders on one drive
    Partition: Set of slices
    Cluster: Set of partitions

  - **Device Overflow**
    Disk full conditions avoided by continuing the write to another device

- **Disk flawing**
  - **Spares cylinder**
    During disk formatting, bad sectors are flagged so data will not be written to these areas, but to the spare (reserved) areas instead; transparent to user

UNICOS File System Enhancements

- **Improved allocation**
  - **Multiple physical blocks per logical block**
- **Clustering of disk partitions**
  - **Disks can be divided up into separate logical devices**
  - **Device Overflow**
- **Disk flawing**
  - **Spares cylinder**

Other UNICOS Enhancements

- **Networking (TCP/IP, USCP)**
  Since Cray computers work with other computers (Network Gateways), they need the protocols to allow them to communicate with each other.

- **Multiprocessing**
  Code needed to be added take advantage of the multiple CPUs in Cray computers.

- **Batch Processing (NQS)**
  Since standard UNIX is interactive-oriented, a facility was necessary to allow batch processing which best suits Cray computers which are primarily batch-oriented. A new option to NQS is a remote queueing system (RQS) which allows a Cray system running NQS to route batch jobs to other Cray systems on the network.

- **Cray FORTRAN (CFT, CFT2, CFT77)**
  Since FORTRAN is the most popular compiler on the Cray, it needed to be enhanced to work efficiently with UNICOS.

- **Accounting**
  Standard UNIX does not provide the usage accounting levels required by Cray customers, so this area has been enhanced.

- **Security**
  Standard UNIX does not meet the "Orange Book" security levels required by the federal government, so security has been enhanced to meet those needs.

Other enhancements to UNICOS

- Networking (TCP/IP, USCP)
- Multiprocessing
- Batch Processing (NQS)
- Cray FORTRAN (CFT, CFT2, CFT77)
- Accounting
- Security

UNICOS 2.0 IMPROVEMENTS

- Batch enhancements
- SCP support for front-end stations
- File system improvements
- Job/process recovery
- Symbolic debugging enhancements
- On-line diagnostics
- New manual for system administrators
- New manual for describing UNICOS internals
- System activity report utility for CRAY X-MP and CRAY-1 computer systems
- Kernel performance improvements on CRAY X-MP computer systems
- Math library tuning on CRAY-2 computer systems
- Flowtrace support added to CRAY-2 computer systems
- UPDATE and SEGLDR ported to CRAY-2 computer systems
- FTREF ported to CRAY X-MP and CRAY-1 computer systems

UNICOS 3.0 IMPROVEMENTS

- TCP/IP improvements
- USCP improvements
- System activity report improvements
- X-window system
- Multi-groups
- UNIX V.3 system calls
- Swapping and scheduling improvements
- Jobs concept
- Jobs accounting
- UNIX V.3 signals
- Job recovery
- NQS logfile management
- NQS extended accounting
- New QMGR commands
- UNIX V.3 command and library enhancements
- CPU targeting
- Pure data unformatted files
- Barrier synchronization
- Line and screen editors (ex and vi)
- HSX support (X-MP, Y-MP and CRAY-1 with IOS)
- Tape support (X-MP, Y-MP and CRAY-1 with IOS)
- dbx symbolic debugger (X-MP, Y-MP, and CRAY-1 with IOS)
- X-MP/14se support
- Hardware performance monitor support (X-MP and Y-MP)
- Secondary data segments (SDS) (X-MP, Y-MP and CRAY-1 with IOS)
- Real-time system capability (X-MP, Y-MP, and CRAY-1 with IOS)
- Boot-time configuration (X-MP, Y-MP and CRAY-1 with IOS)
- Multi-task debugger yadb (CRAY-2)

UNICOS 4.0 IMPROVEMENTS

- Support for CRAY Y-MP and CRAY X-MP EA systems
- Support for DS-40 Disk Subsystem
- Support for CNT LANlord network connection
- Enhancements to High-speed External Channel (HSX) communications channel support
- New security feature (available with UNICOS 5.0 for CRAY-2)
- Tape subsystem enhancements
- Network File System (NFS) support for CRAY-2
- Network Queueing System (NQS) enhancements
- Accounting enhancements
- New Fortran compiler interface commands
- Extensions to job and process recovery
- Microtasking enhanced and now availabe on CRAY-2
- Assign command available
- Support for temporary files
- Dump and restore backup utilities available
- Logical device cache on SSD supported
- File System Switch (FSS) now available on all systems
- Enhanced C compiler
- Enhanced or new system calls, commands and library routines
- Support for MIT X Window System version 11
- Enhanced Guest Operating System (GOS) and migration utilities
- Enhanced UNICOS Station Call Processor (USCP)
- Enhanced TCP/IP

UNICOS 5.0 IMPROVEMENTS

The primary purpose of UNICOS 5.0 is to provide COS functional equivalency. COS functional equivalency allows users of COS to perform their work under UNICOS with at least the same degree of flexibility and control that they enjoyed with COS. UNICOS 5.0 includes numerous other features.

The major software enhancements to support new hardware include the following:

- DS-40 Disk Subsystem Daisy-chaining
- NSC HYPERchannel connection
- Operator workstation
- Disk striping for CRAY Y-MP and CRAY X-MP EA systems
- CNT LANlord connection
- UltraNet support
- Disk offloading support

The major new or enhanced operating system software includes:

- Tape subsystem
- File system monitor
- User Database (UDB)
- Network Queueing System
- Share scheduler
- User-level disk striping
- Resource limits
- crayperf
- File system performance improvements
- Real-time
- SSD SDS administration and scheduling
- New or changed adminstrator commands
- Accounting
- UNICOS multi-level security
- Data migration
- Remote Queueing System
- Restrict default path
- Temporary Files
- /proc
- Memory scheduling
- Guest Operating System (GOS)
- New user commands
- New or changed system calls
- IOS

The enhancements to UNICOS language processors includes:

- CFT77 3.0
- CFT 1.15 Bugfix
- Pascal 4.0
- Cray Standard C 1.0
- Cray Allegro Common Lisp 1.0
- CFT2 5.0
- Cray C 4.1
- CAL version 2, release 3.2
- Cray Ada 1.0

Major library enhancements include:

- Scientific library (libsci)
- User striping support
- Autotasking
- Multi-threading I/O improvements
- Utility library (libu)
- I/O library (libio)
- Network libraries
- C library (clib)
- Targeting
- Fortran library (libf)
- Multitasking library routines
- Facilities to convert foreign data

Enhancements to software products include:

- Debuggers (adb, CDBX, dbx, DRD, DDA, DEBUG)
- Cray Simulator (CSIM)
- SUPERLINK 2.2
- UltraNet software support
- On-line diagnostics

5.0 UNICOS PRODUCT SET
(Main components of standard UNICOS)
(release versions vary by system)

- UNICOS 5.0 Operating System
- Cray C
- CFT77
- PASCAL
- CAL Version 2
- CFT2 (CRAY-2 systems only)
- NQS (and RQS, if ordered)
- SEGLDR
- CDBX and other debuggers
- USCP
- TCP/IP (license required)
- Network File System (NFS)
- Fortran, C, and Pascal Library Support
- Cray Simulator (CSIM)
- On-line diagnostics

SECTION 8

Cray Product Familiarization

THE IOS OPERATING SYSTEM

Cray Research, Inc. Software Training

THIS PAGE IS INTENTIONALLY LEFT BLANK

INPUT/OUTPUT SUBSYSTEM (IOS) OPERATING SYSTEM

The I/O Subsystem Operating System is a separate operating system which directs the operation of the IOS hardware and works in conjunction with either COS or UNICOS on CRAY-1 and X-MP systems. In fact, it is considered to be a subset of the COS or UNICOS operating systems when used on those machines.

It performs I/O between central memory and peripheral devices attached to the IOS. IOS software:

- Drives the disk storage units
- Drives the front-end channels
- Drives the user channels
- Drives the IBM tape drives - Block Mux (BMX)
- Performs master operator station functions using the expander chassis equipment

The IOS Operating System also deadstarts the mainframe and is used to perform offline diagnostics on the mainframe.

IOS OPERATING SYSTEM COMPONENTS

Like COS or UNICOS, the IOS Operating System has a central control program. It is called the **Kernel**. A copy of the Kernel executes in each of the I/O processors with only minor modifications, based on the function of each processor.

Like COS's tasks or UNICOS's utilities, the IOS Operating System has a group of executable programs called **overlays**. When an overlay is called by a specific I/O processor (e.g., the Disk I/O Processor (DIOP) calling an overlay to do disk I/O), that program is loaded into the local memory of the calling processor and begins to execute. Other overlays called later simply overwrite the memory area where the previous overlay was.

Copies of all overlays and a master copy of the Kernel are kept in Buffer Memory.

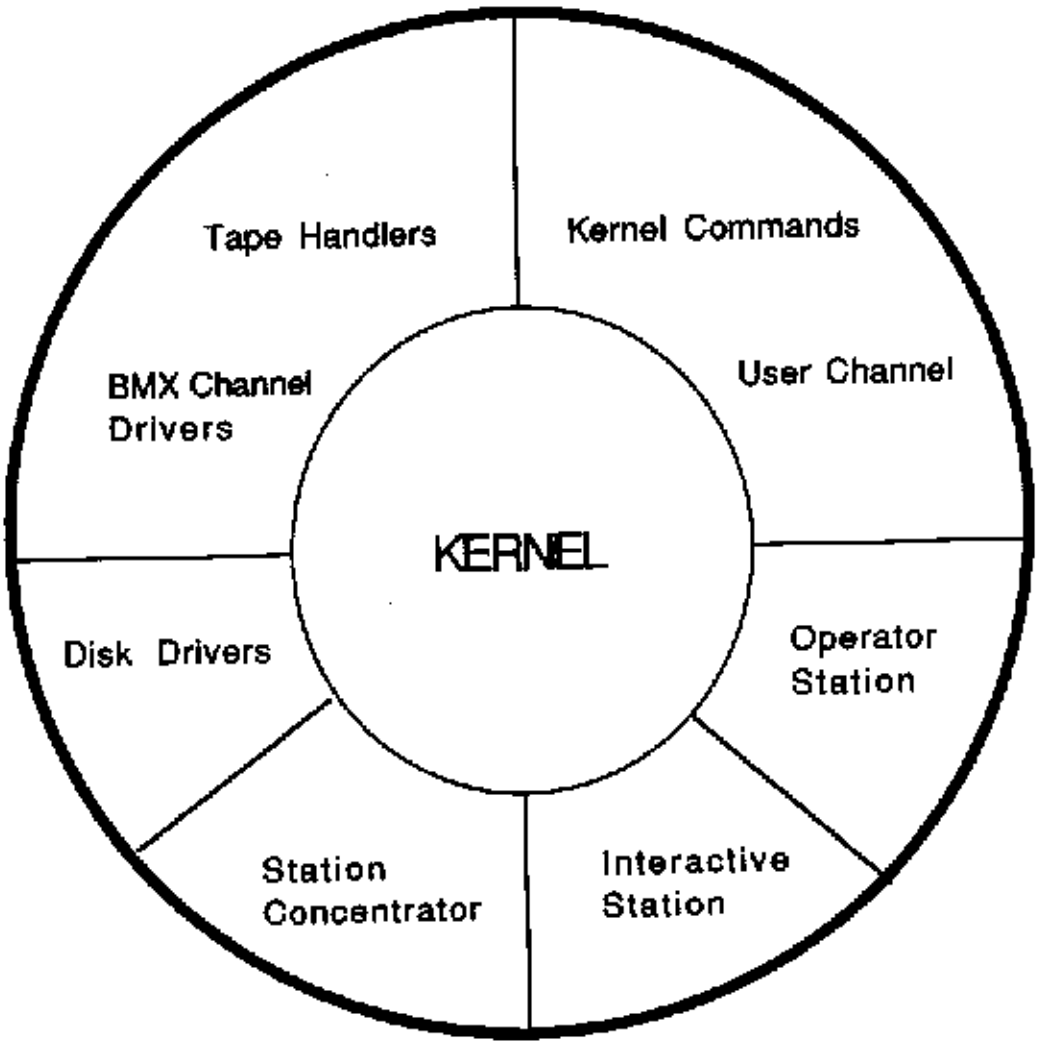
Types of software programs within the IOS Operating System include:

- Station Concentrator - receives and routes data and messages from front-ends to and from the mainframe
- IOP Station and Interactive Station - these two software elements allow operators to oversee and control system operations and to run their own programs interactively on the mainframe
- Tape and BMX Driver - these programs interpret tape read or write requests and control the tape channels
- Kernel Overlays - Individual subprograms to do specific work
- Disk - program to interpret disk read or write requests and control the disk channels

Another important part of IOS software is the packet driver which sends messages across the 6Mbyte/sec. channel on the Master I/O Processor to the mainframe. These messages contain information regarding the sending and receiving of data over the high speed channels (100 Mbyte/sec.).

IOS software is written in an assembler language called A-Processor Middle Language (APML). This group of instructions is not used by any other Cray hardware.

IOS System Components



THIS PAGE IS INTENTIONALLY LEFT BLANK

SECTION 9

Cray Product Familiarization

Communications and Networking

Cray Research, Inc. Software

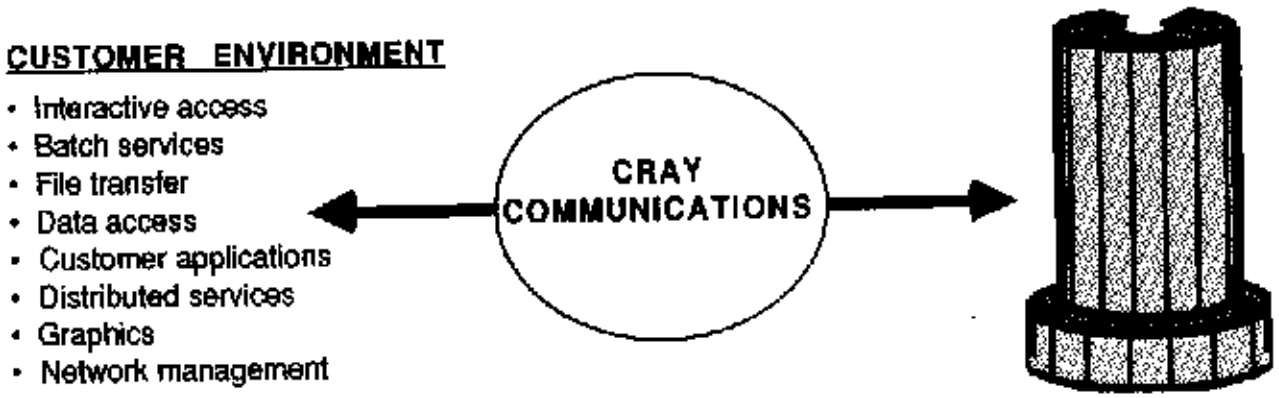
THIS PAGE IS INTENTIONALLY LEFT BLANK

CRAY NETWORKING PHILOSOPHY

Cray systems do little good if users don't have easy access to the systems' power. By providing an organized network of software and hardware, many users can share the system.

Communication strategy at Cray Research, Inc. is to provide Cray system functionality to the end user. The intent is to fit Cray systems within existing and future customer environments. These environments include single or multiple Cray systems, other vendors' mainframes, minicomputers, workstations, and devices capable of high-speed data transfer.

To provide this, Cray currently offers two main communication environments: Cray station and an industry standard, TCP/IP. Others are available and still others are evolving.



CRAY NETWORKING HARDWARE OPTIONS

Network hardware allows Cray computer systems to communicate with virtually any mainframe, minicomputer, workstation, or high-speed storage device. Many price/performance options are offered by Cray Research and third party vendors.

The main ones utilized in Cray networking schemes are listed below and described in further detail on the following pages.

Front-end Interface

NSC HYPERchannel

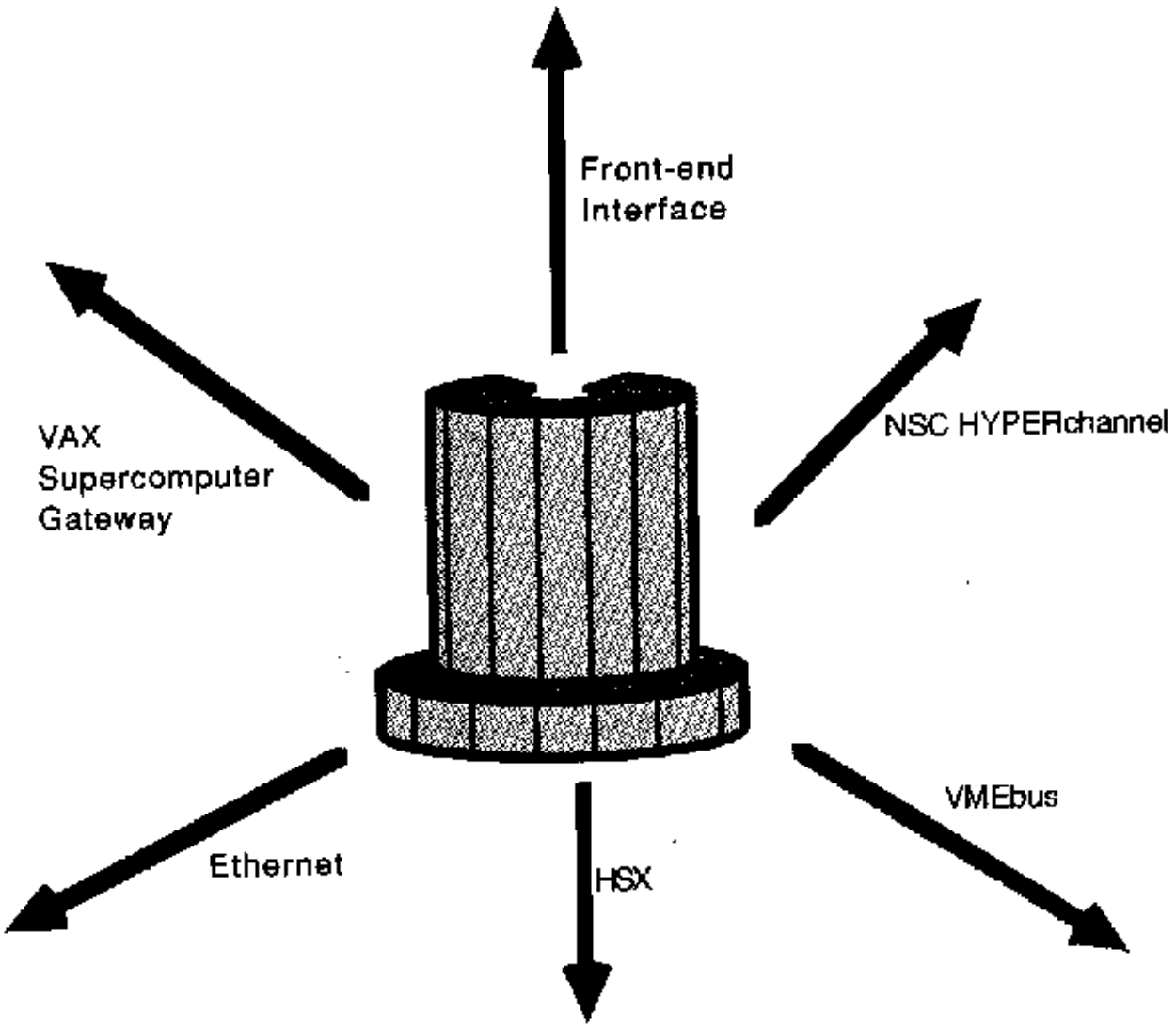
VMEbus (FEI-3)

HSX

Ethernet

VAX Supercomputer Gateway

CRAY
NETWORKING OPTIONS
(HARDWARE)



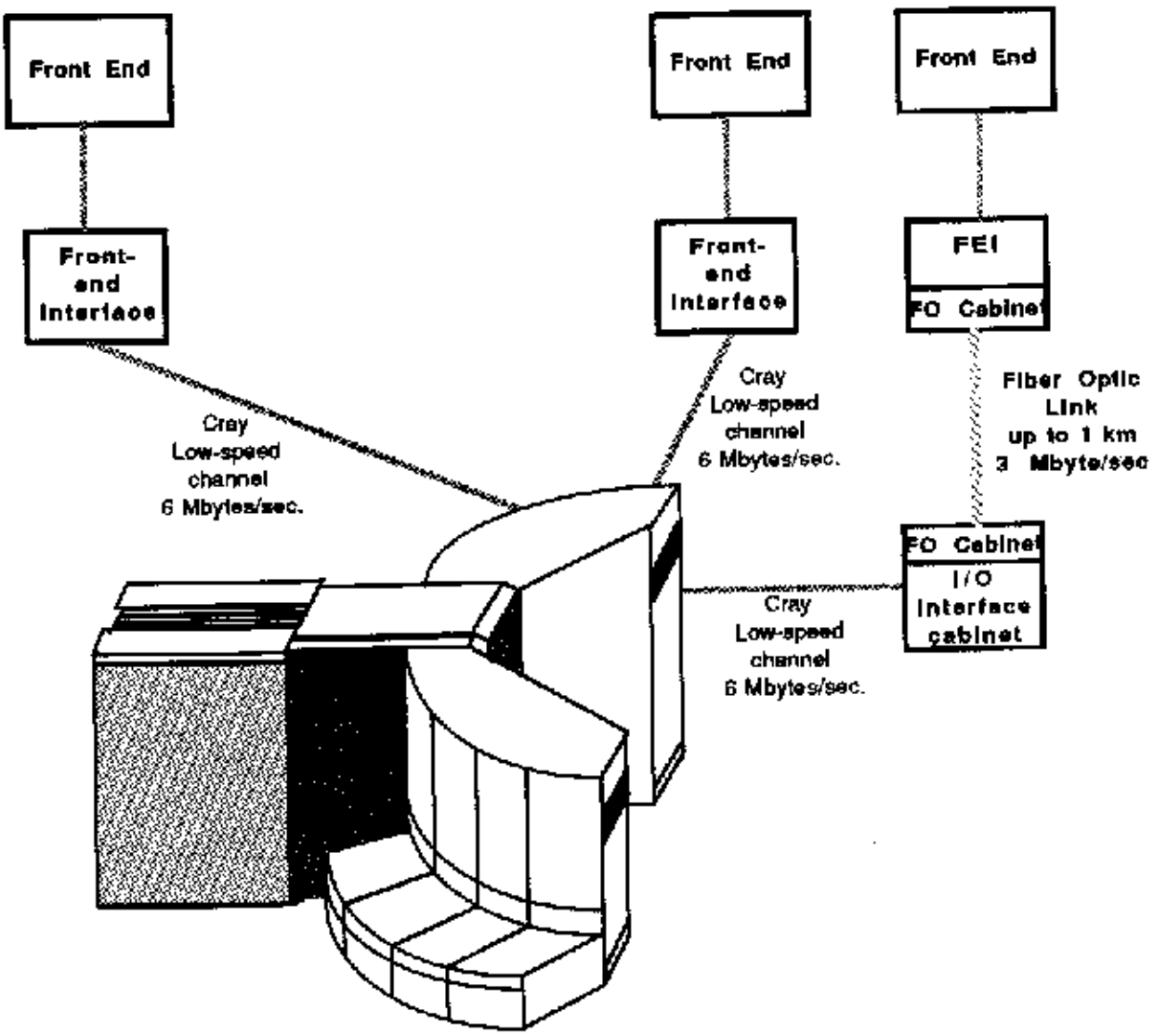
FRONT-END INTERFACE

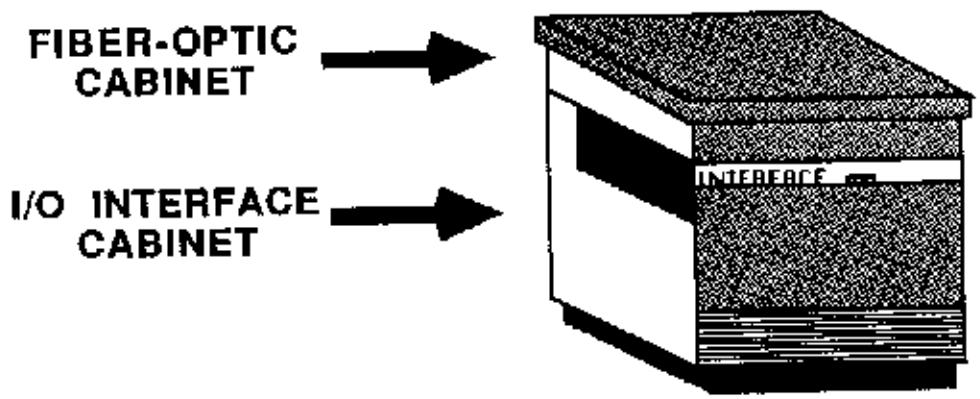
The FEI is the "traditional" connection between a Cray computer and a front-end computer. It is a point-to-point connection; that is, one cabinet with specific Cray modules in it interfaces to a specific front-end computer. The FEI connects either directly to a Cray channel or through MIOP in the IOS. FEIs compensate for differences in channel widths, word size, logic levels, and control protocols between other vendors' equipment and Cray computer systems.

Data transfer rates are from 1.5 Mbits/sec. to 3 Mbits/sec., depending on the front-end channel speed. Maximum cable length is 70 ft. An optional fiber optic link allows an FEI to be separated from a Cray system by up to 1000 meters and provides complete electrical separation of connected devices. The fiber optic modules are housed in a "dormer" on top of the basic interface cabinet.

Some of the front-end computer systems FEIs have been developed for include IBM, DEC, CDC, UNISYS, and Honeywell.

Front-End Interfacing





NSC HYPERchannel

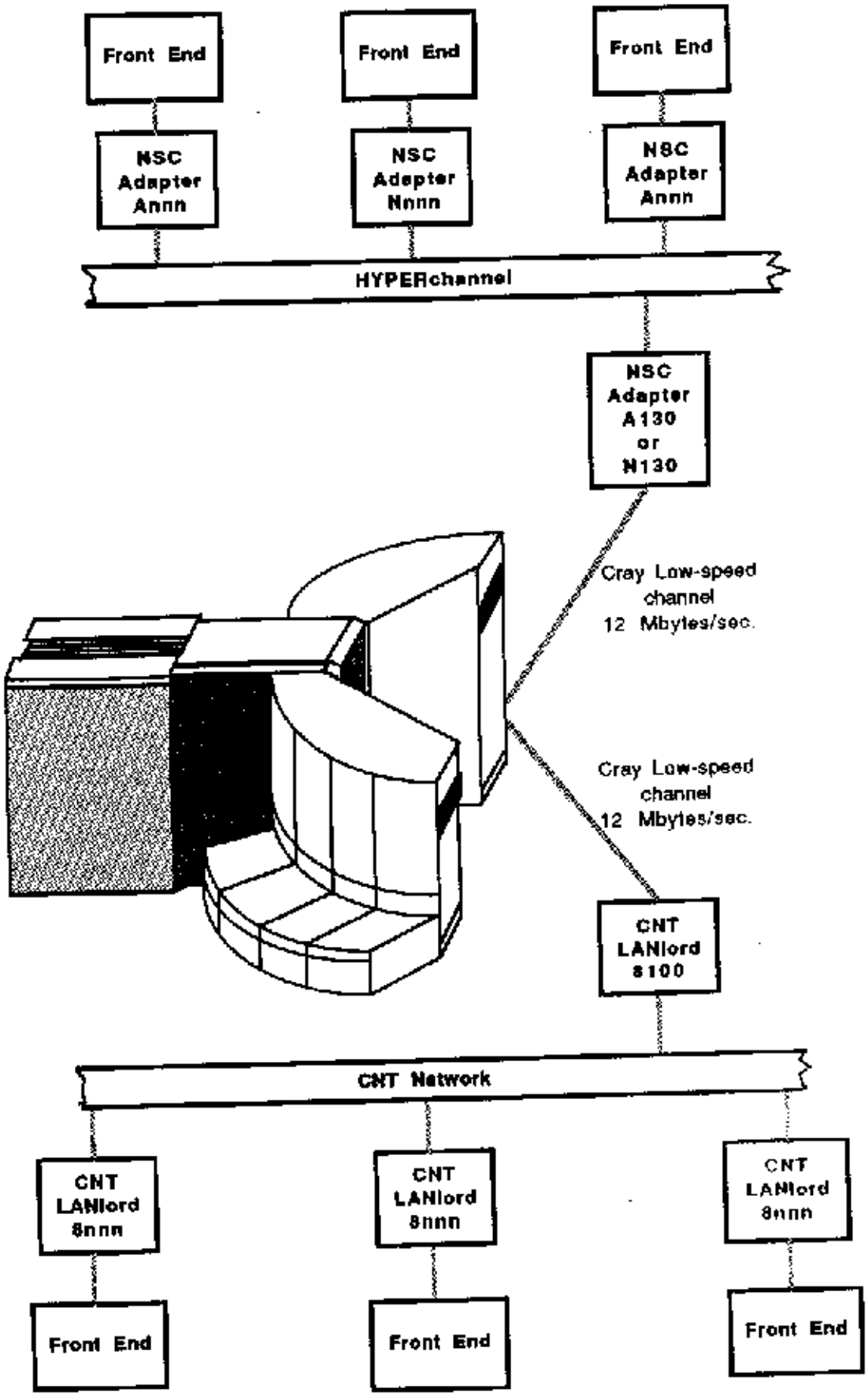
The Network Systems Corporation (NSC) HYPERchannel has been one of the most common ways to connect a Cray computer system to the outside world for many years and it continues to be an important networking option.

An NSC A130 or N130 adapter connects to the Cray low-speed channel and communicates with computer systems connected to other A- or DX-series (also referred to as N-series) adapters via a coaxial trunk. NSC's HYPERchannel-50 (50 Mbits/sec.) network is supported by both A-series and DX-series. NSC's HYPERchannel-100 (100 Mbits/sec.) is supported by the DX-series only.

CNT LANlord

A product similar in scope to NSC HYPERchannel is Computer Network Technology Corp.'s (CNT) LANlord. It provides a high-speed communication interface between Cray computer systems and other hosts on a local area network (LAN).

NSC HYPERchannel and CNT LANlord



VMEbus (FEI-3)

The Versabus-Modular-Eurocard (VME) is a modified version of Motorola's proprietary VersaBus using the Eurocard standard form factor (for board dimensions and connectors). This "standard" allows a variety of boards to fit into a VMEbus backplane in a computer and do all kinds of work.

Cray used this format to design a new communications connection to workstations that utilize the VMEbus backplane. This Cray product is a two-board set called the FEI-3. Workstations supported or planned for support include Sun 3 and 4, Motorola, Silicon Graphics, and Apollo.

These boards fit into a workstation's backplane. Each board has an edge connector that connects to a Cray low-speed channel. One board/channel is input, the other is output. The FEI-3 board set contains no CPU or intelligence of its own. The boards for all workstations are identical but must be configured via jumpers and switch settings for each workstation model. In addition, the workstation vendor must supply appropriate driver software for their operating system.

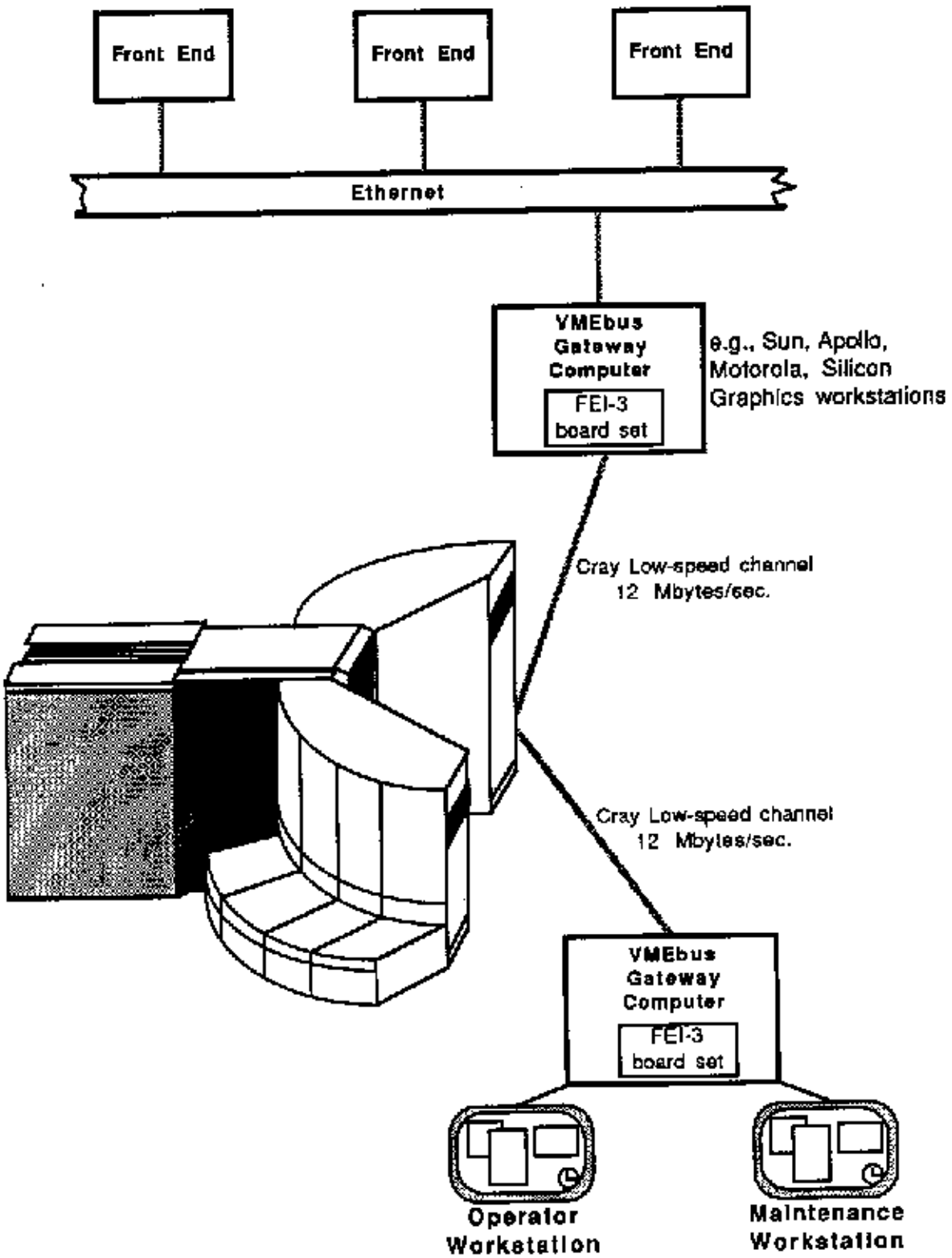
The workstations containing these boards become the gateway to other networks such as Ethernet. The FEI-3 looks like an NSC A130 to the Cray software and is capable of many communications protocols including (U)SCP and TCP/IP. The FEI-3 is also less expensive than NSC HYPERchannel and, in fact, is the least expensive way to connect a Cray to the outside world, although it may not be the most efficient way. It requires the VMEbus minicomputer containing the FEI-3 boards to be dedicated as the gateway.

The FEI-3 is also being used with the new Operator and Maintenance Workstations.

ETHERNET

Ethernet, a product of Xerox Corp., is the most commonly used industry standard Local Area Network (LAN). It uses coaxial cable or twisted-pair to connect computers in a small area (usually no more than one building). From there, an Ethernet network can connect to other Ethernet networks. Ethernet can also connect to Cray computer systems, but not directly. They must feed into configurations such as HYPERchannel or VMEbus gateways. Its channel speed is rated at about 10 Mbits/sec.

FEI-3 (VMEbus)


Front End
Front End
Front End
Ethernet
VMEbus Gateway Computer
FEI-3 board set
e.g., Sun, Apollo, Motorola, Silicon Graphics workstations
Cray Low-speed channel 12 Mbytes/sec.
Cray Low-speed channel 12 Mbytes/sec.
VMEbus Gateway Computer
FEI-3 board set
Operator Workstation
Maintenance Workstation

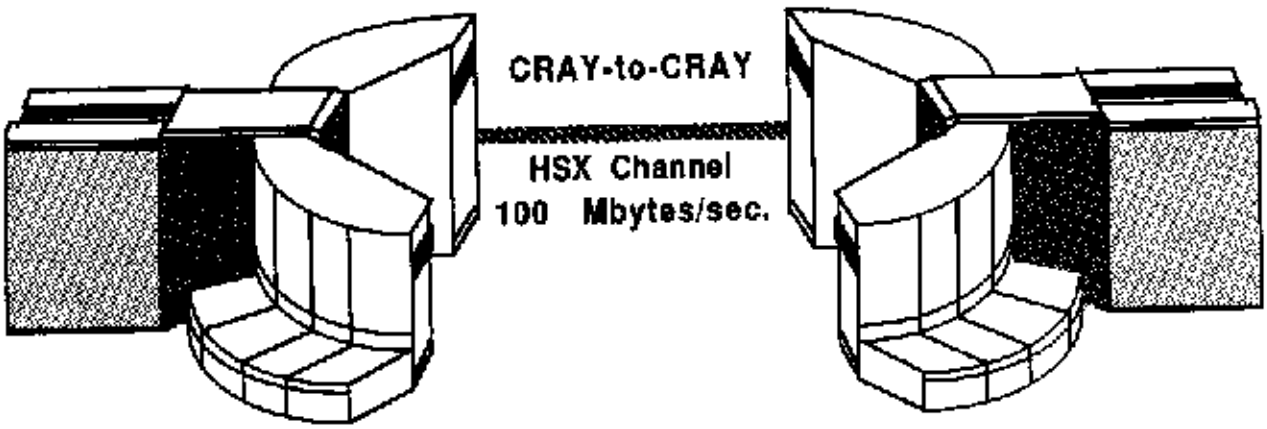
HSX

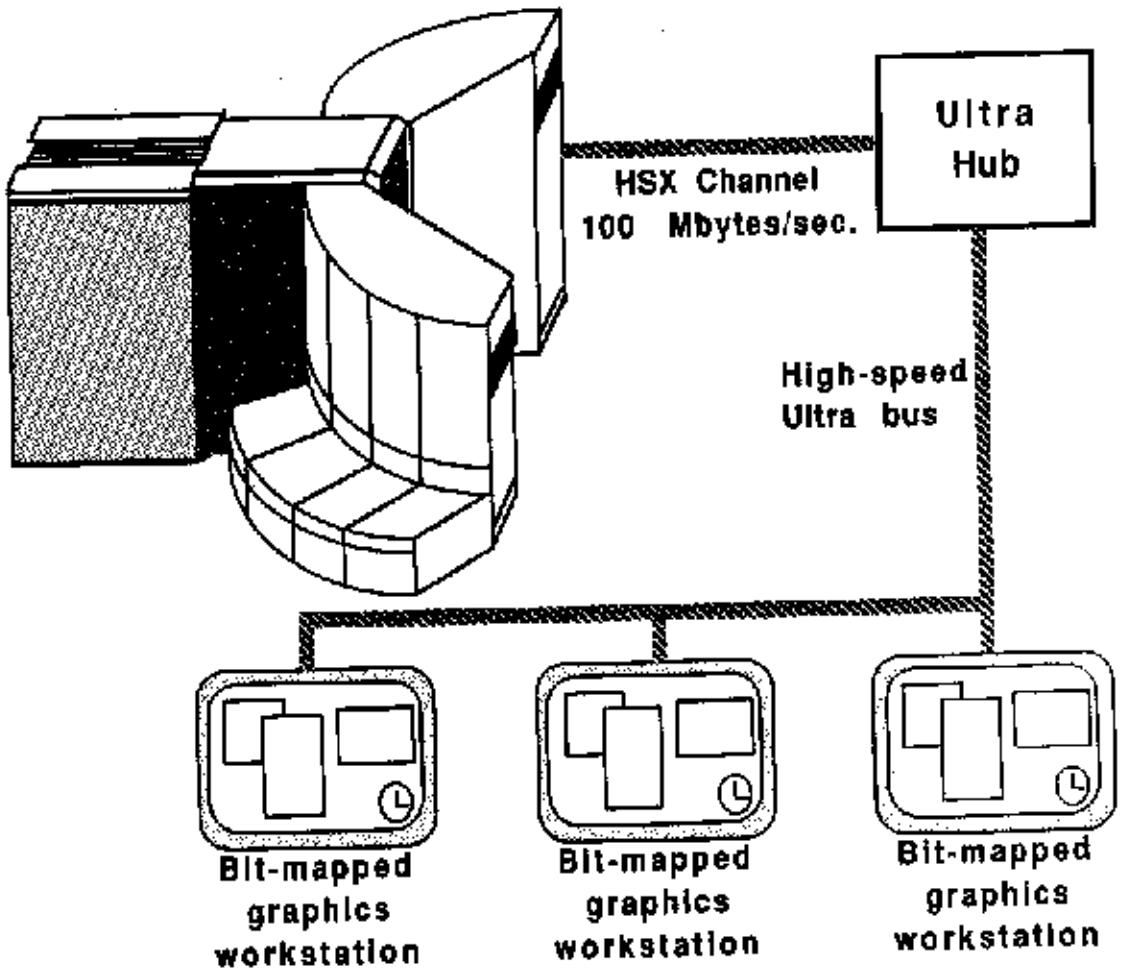
The High-speed External (HSX) communications channel provides full-duplex (both directions at the same time) point-to-point communications between two Cray computer systems or between a Cray computer system and a user-supplied device such as a high-speed graphics output device. It operates at speeds up to 100 Mbytes/sec.

The Cray-to-Cray connection uses the TCP/IP protocols and can send data at speeds approaching 170 Megabits/sec. with UNICOS 5.0.

The HSX connection to high-speed graphics devices is still evolving. An example is a connection to an UltraNet system. UltraNet is a product of Ultra Network Technologies. The UltraNet network will support Convex, Alliant, Sun, and Silicon Graphics computer systems, as well as Cray computer systems.

HSX


CRAY-to-CRAY
HSX Channel
100 Mbytes/sec.


Ultra
Hub
HSX Channel
100 Mbytes/sec.
High-speed
Ultra bus
Bit-mapped
graphics
workstation
Bit-mapped
graphics
workstation
Bit-mapped
graphics
workstation

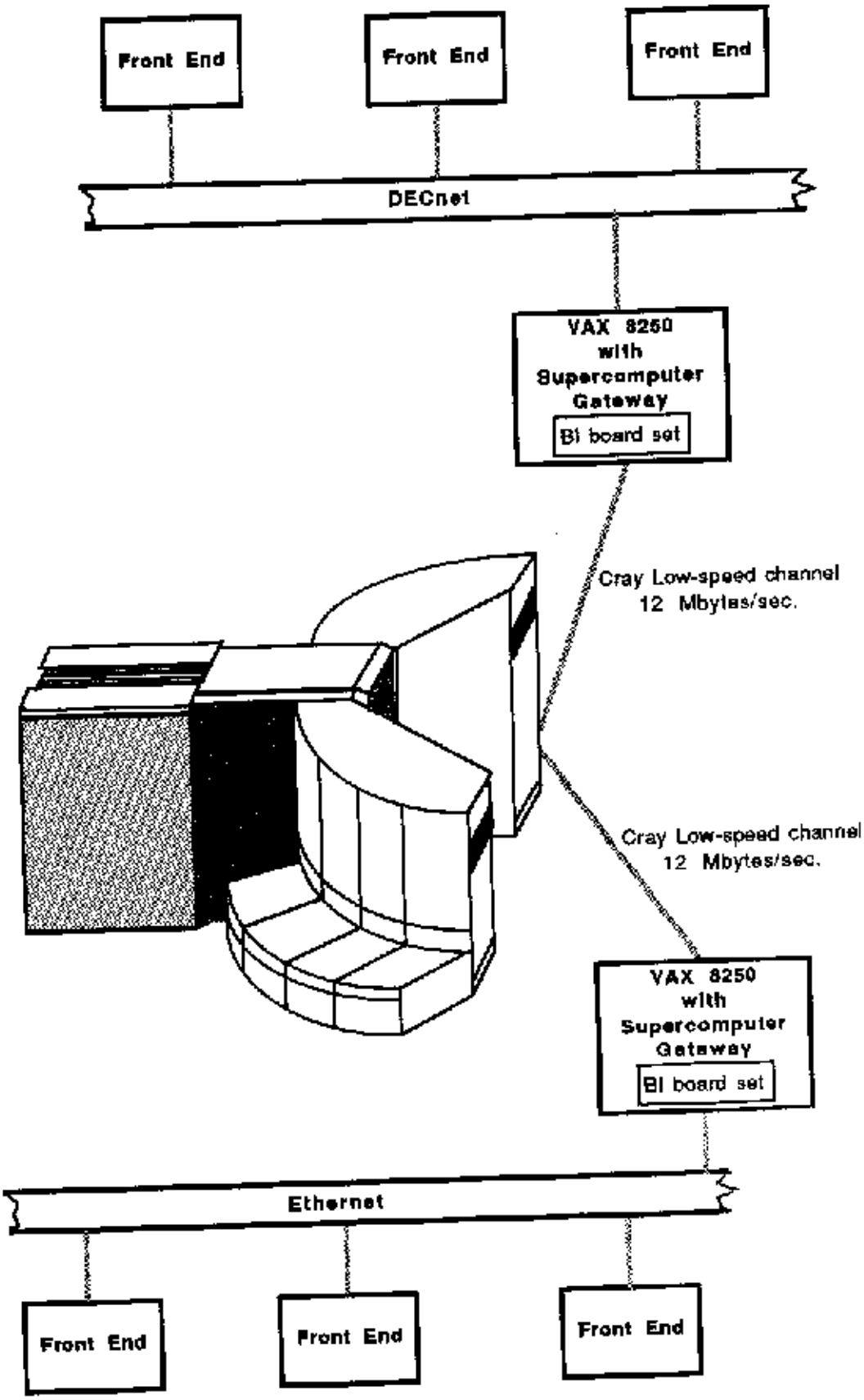
VAX Supercomputer Gateway

The VAX Supercomputer Gateway is a Digital Equipment Corp. product. It consists of a VAX 8250 computer system with a Backplane Interface (BI) board set for connection to a pair of Cray Low-speed channels in a manner similar to the FEI-3. Two Crays may be connected to a single supercomputer gateway system.

The main benefits of the gateway include improved throughput (performance 2-3 times that of a VAX FEI) and lowered cost by allowing many other VAX computers (on a network such as Ethernet or DECnet, for example) to connect to a Cray through just one means.

Protocols supported include DECnet and TCP/IP.

VAX Supercomputer Gateway



CRAY NETWORKING SOFTWARE OPTIONS

Network software allows Cray computer systems to communicate with virtually any mainframe, minicomputer, workstation, or high-speed storage device via network hardware connections.

The main communications software utilized in Cray networking schemes are listed below and described in further detail on the following pages.

Station Software (SCP, USCP)

Transmission Control Protocol/Internet Protocol (TCP/IP)

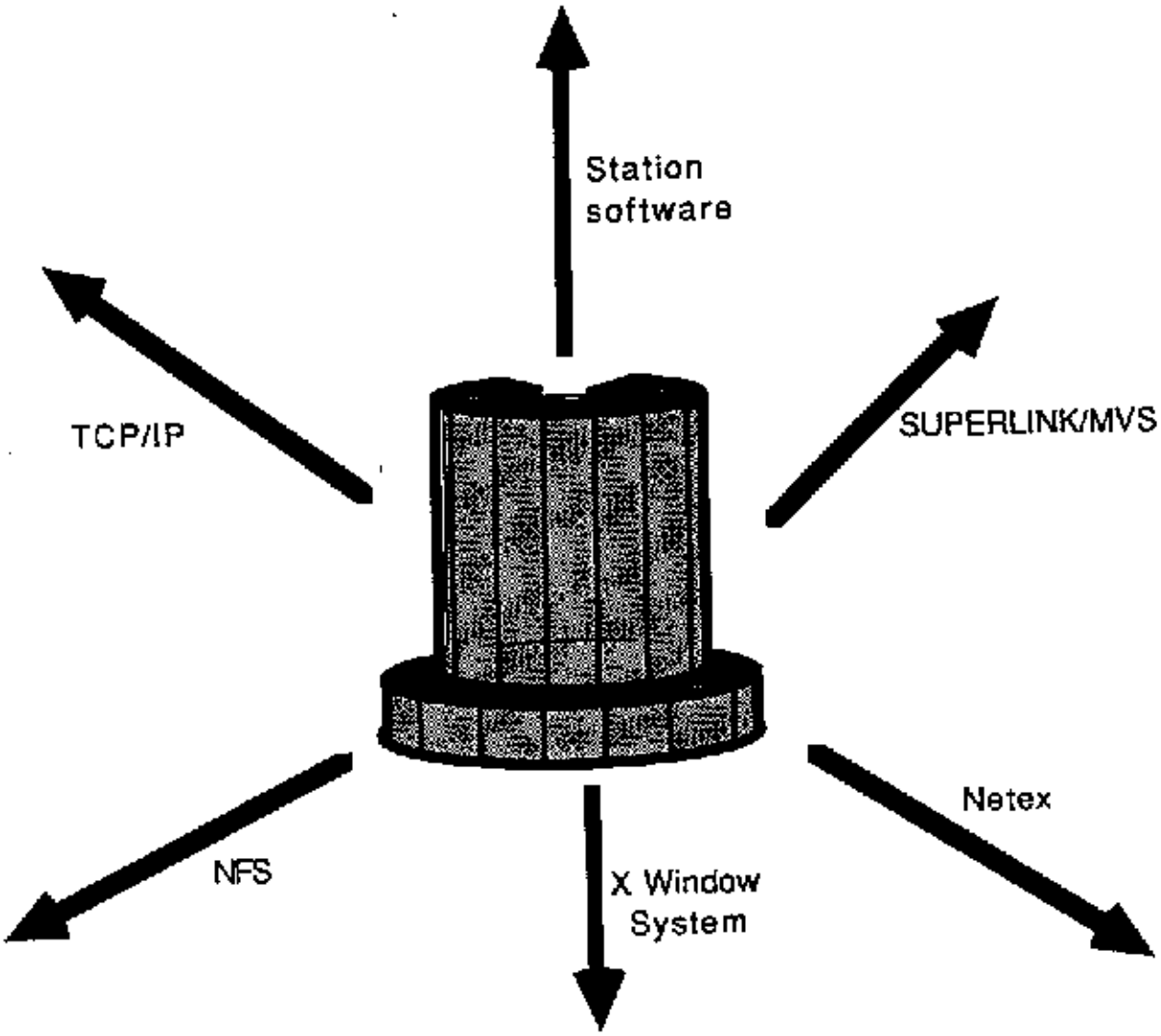
SUPERLINK/MVS

Netex

Network File System (NFS)

X Window System

CRAY NETWORKING OPTIONS (SOFTWARE)



CRAY STATION SOFTWARE FEATURES

The Station Call Processor (SCP) and UNICOS Station Call Processor (USCP) are Cray's traditional software protocols to link to front-end computers.

Software specific to a front-end computer and executing in that front-end passes user jobs to the Cray and assimilates output from the Cray. SCP/USCP in the Cray interprets and executes the commands received from the front-end. Station software is basically a file-passing protocol designed around the idea Cray computer systems are batch-oriented.

Interactive station is also available but is limited in scope and use.

CRAY STATION SOFTWARE SERVICES

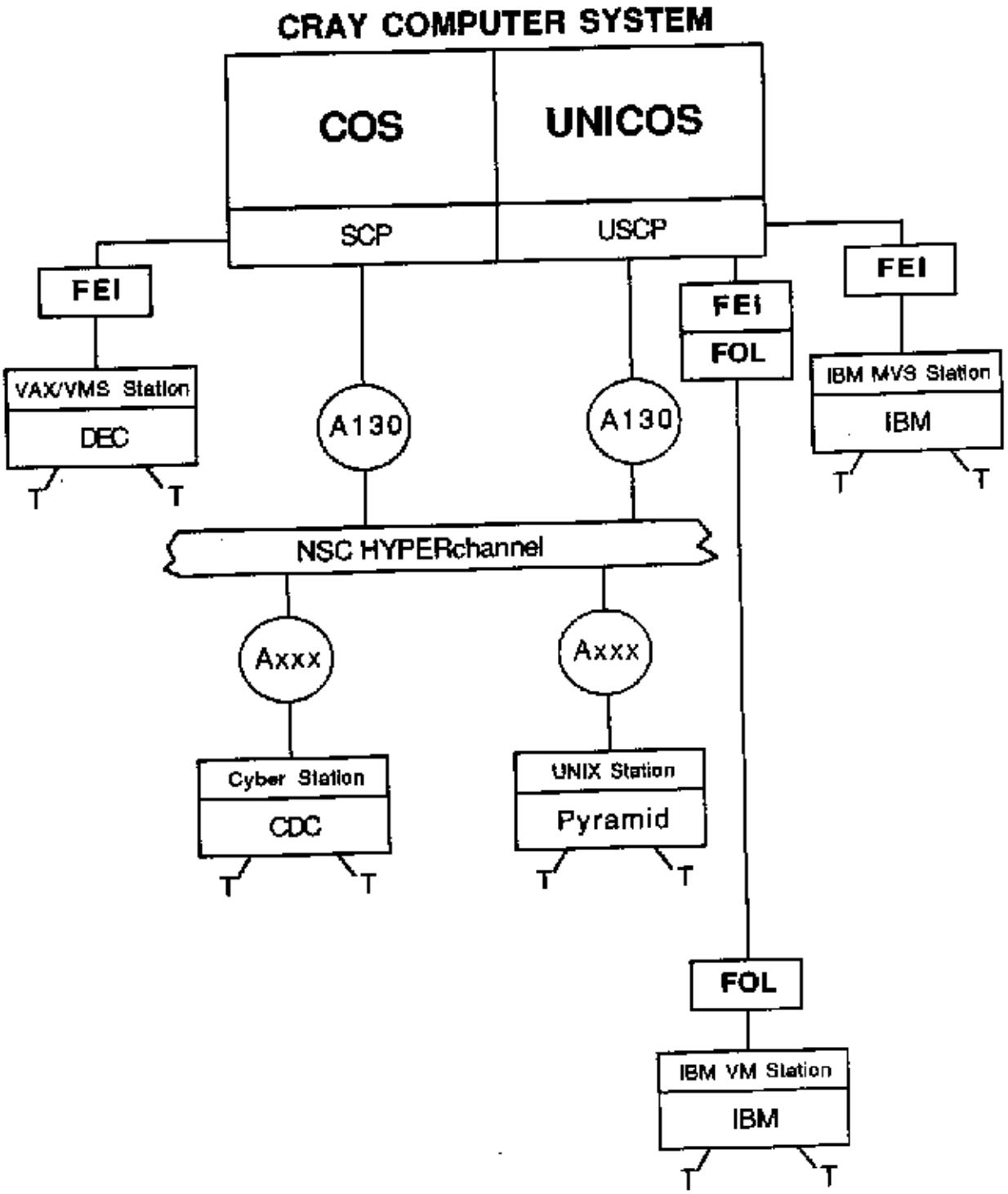
CRAY stations support either Front-end Interface (FEI), FEI-3, or NSC HYPERchannel connections.

These stations are available from CRAY Research:

- IBM MVS JES2/3
- IBM VM
- CDC NOS, NOS/BE, NOS/VE
- DEC VAX/VMS
- APOLLO AEGIS
- UNIX SYSTEM V (VAX, SUN, 3B20)
- SUPERLINK/ISP
- DATA GENERAL AOS

- Third party stations:
  - AT&T UNIX
  - Honeywell GCOS
  - Honeywell MULTICS
  - Sperry SST/1100
  - Pyramid
  - Iris

STATION NETWORK CONFIGURATION EXAMPLE



OPEN SYSTEMS INTERCONNECT (OSI)

Open Systems Interconnect (OSI) is a specification for the design of compatible communications protocols defined by the International Standards Organization (ISO). It is not a product, but rather an attempt at defining a network architecture which could become the template for all new communications protocols. Its aim is simple - communications transmissions between networked computers should not be hampered by proprietary standards.

OSI is intended to address the limitations of current network protocols - TCP/IP specifically. It incorporates characteristics of several existing protocols including TCP/IP, SNA, and X.25 and is organized into software "layers" for modularity and ease of enhancement. Definition of this standard is not complete and will not be for several years because, while the idea is simple, its implementation is complex and requires agreement between vendors on the design and behavior of seven different layers. Once agreement is reached, however, protocols based on the seven layers will be easily enhanced and modular.

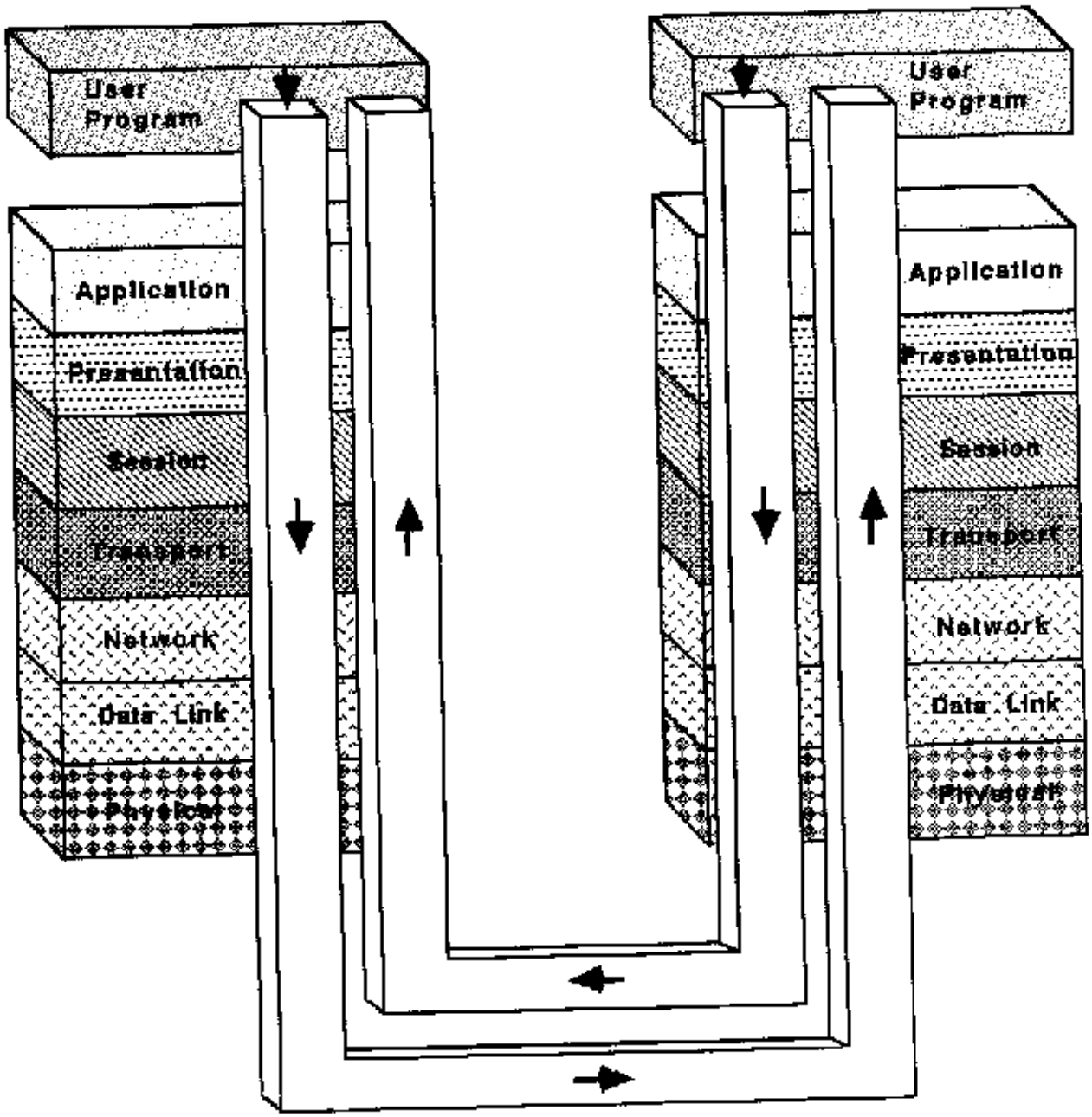
The OSI Reference Model Layers

The seven layer reference model on which OSI is based provides a framework that defines the functions required in a communications transmission between networked computers. In essence, each layer of the protocol on one computer "talks" to the complementary layer on the receiving computer.

The layers are:

7. **Application**
   On each host there is a program that wishes to communicate with its counterpart on the other host. This may be a file transfer utility like **ftp** or a user-written program that includes system calls to network functions.
6. **Presentation**
   This layer performs common data manipulation functions, such as compression or encryption.
5. **Session**
   This layer controls the connection of the communicating processes; sets half or full duplex; packet routing (so the right process receives the right message).
4. **Transport**
   This layer controls the quality of the transmission between the communicating hosts; flow control, transmission speed adjustments.
3. **Network**
   This layer controls the routing of information through the network; controls congestion.
2. **Data Link**
   This layer packages data for transmission and unpackages them for receipt. It divides the bit stream into "frames" and adds acknowledgements.
1. **Physical**
   This layer drives the physical network components connecting the hosts.

The Open Systems Interconnect Model (OSI)



TCP/IP FEATURES

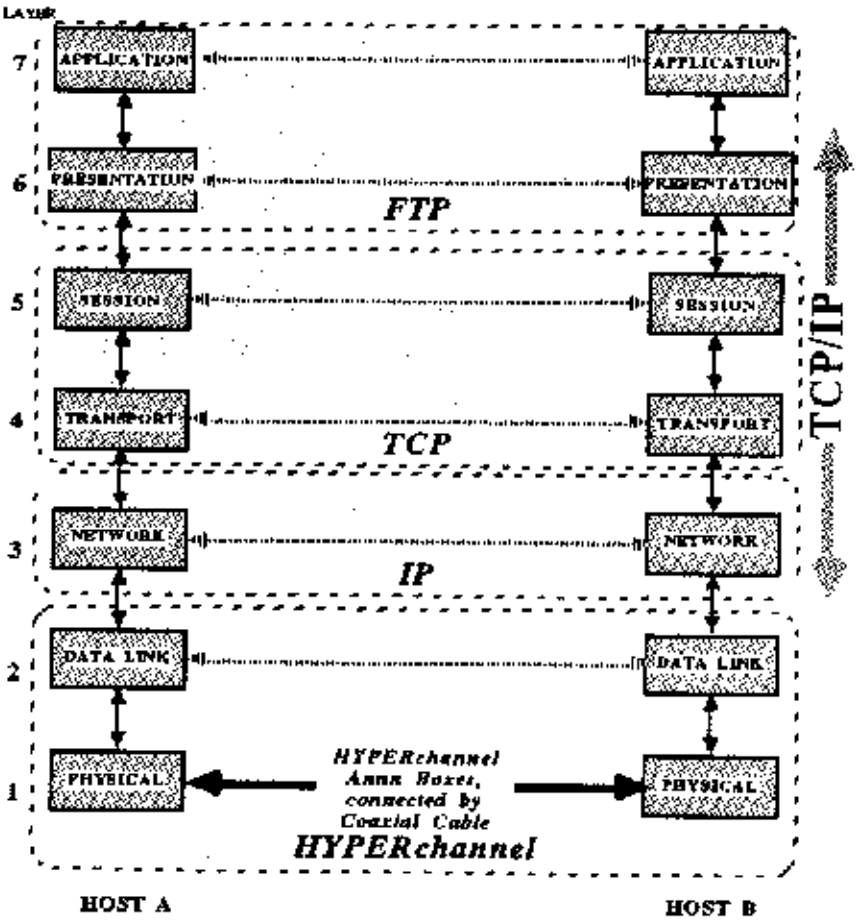
Transmission Control Protocol/Internet Protocol (TCP/IP) is a Department of Defense networking protocol designed to be a "standard" which would allow many different types of computers to communicate with each other easily. It is one of the ancestors of the OSI model. TCP/IP is common in the UNIX world because it is included in the popular Berkeley version of UNIX. Any computers which have TCP/IP software and are on interconnecting networks can communicate with each other.

TCP/IP is available under license with UNICOS and provides a high-performance link that allows Cray computer systems to communicate with other vendor systems that have implemented this open system standard. It allows users to transfer and copy files between their local computer systems and the Cray computer system. TCP/IP file transfer utilities are particularly powerful because they provide automatic authorization on remote hosts and allow file transfer between all hosts that support TCP/IP, regardless of the operating system.

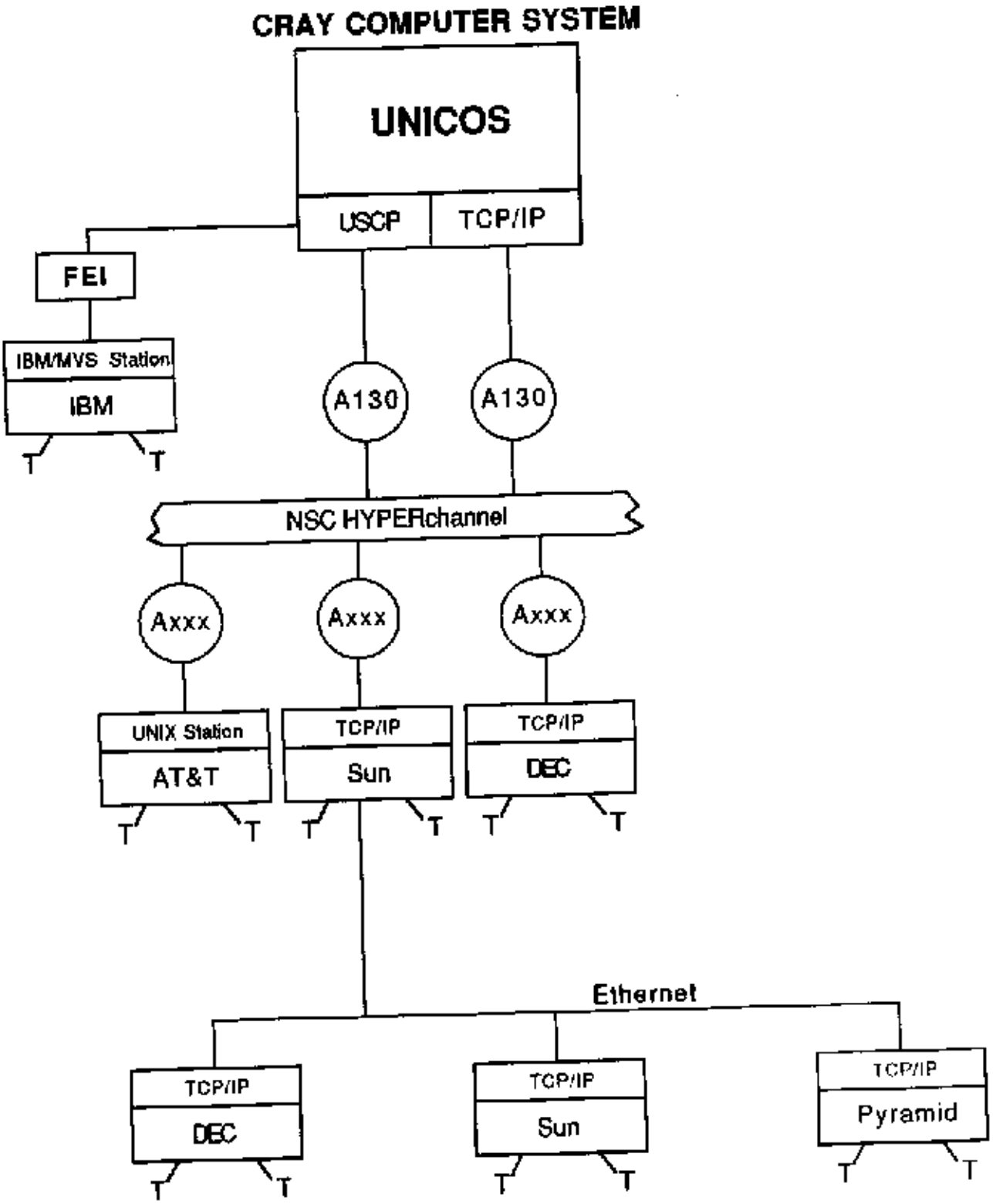
The following utilities are a sample of TCP/IP user facilities:

rlogin provides direct remote login between UNIX systems and CRAY systems
telnet provides a remote login to a remote host
remsh executes a command on a remote UNIX system
rcp transfers files between UNIX hosts
ftp transfers files between hosts, regardless of host operating system

TCP/IP and the OSI Model



TCP/IP CONFIGURATION EXAMPLE



SUPERLINK/MVS

SUPERLINK/MVS is a high-performance link between a Cray computer system and an IBM or compatible system. It is used in combination with Cray Research's IBM MVS Station software to offer many more features.

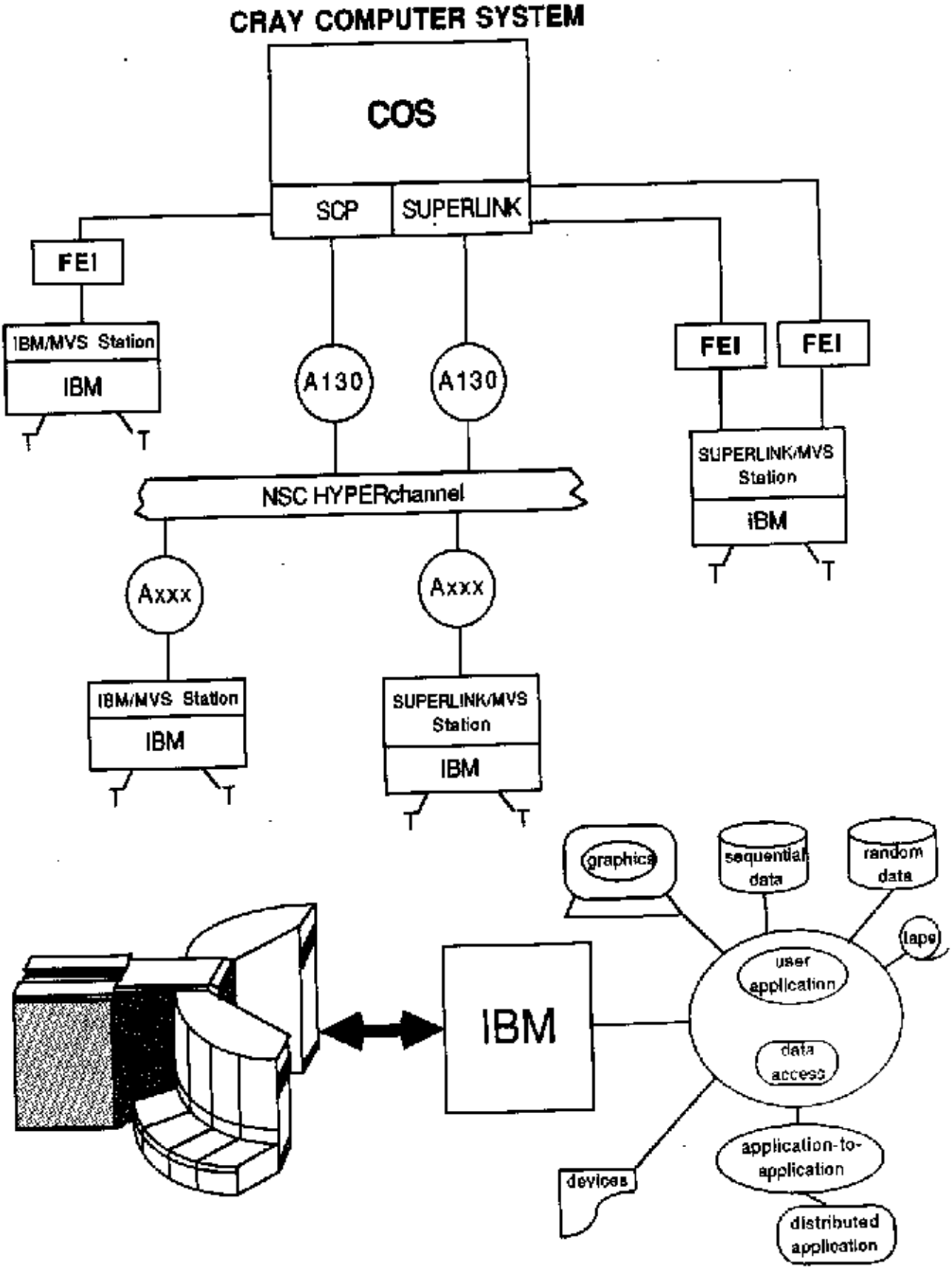
SUPERLINK/MVS provides a sophisticated software link between UNICOS or COS and MVS operating systems. One of its chief features is to provide random and sequential record-level access to the IBM MVS environment.

It also:

- Supports formatted/unformatted I/O, buffered I/O, and assembly language routines
- Is an alternative to staging datasets (i.e., having to copy a complete dataset to the Cray before work can be done on it)
- Handles translation of dataset formats between UNICOS/COS and MVS
- Passes MVS dataset data directly to user memory, saving both disk space and time
- Provides access to datasets too large for Cray mass storage
- Provides several data management services such as dataset recovery, accounting, and security
- Is based on the OSI architecture, with modifications to enhance performance
- Allows application-to-application communication (distributed processing) - an MVS process can create a COS process and vice versa; distributed processing is transparent to user
- Works in combinations of HYPERchannel or FEI
- Use of multiple link communication devices (e.g., 2 FEIs from one MVS system)

SUPERLINK/MVS is expected to eventually absorb standard MVS station and the SUPERLINK template will be used to design enhanced station software for other computers.

SUPERLINK/MVS CONFIGURATION EXAMPLE



NETWORK FILE SYSTEM (NFS)

Network File System (NFS) is Sun Microsystem's proprietary file-sharing software. It is designed to make a remote UNIX environment's file system appear to be local on a Cray computer system. It operates under the TCP/IP protocol.

NFS makes any number of remote file systems appear to be mounted locally. In other words, NFS allows files to be accessed with standard UNIX I/O calls, independent of the location of the files. Users and programs see the usual hierarchy of directories and files; they manipulate these files and directories in the usual way, subject to normal permission checks. The fact that parts of their file system may reside on various machines around the network is invisible and of no concern to the average user.

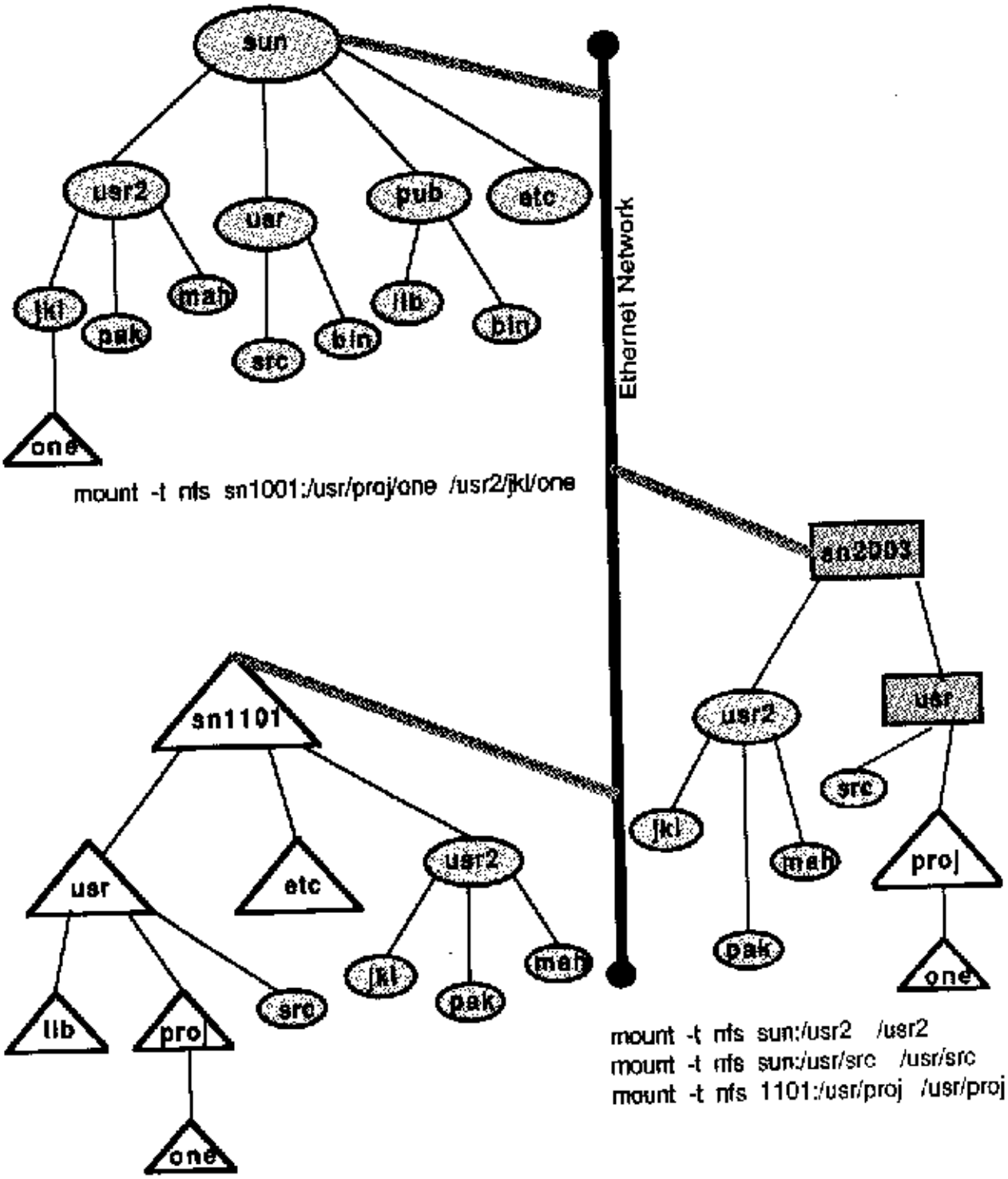
Users are able to get directly to the files they want without knowing the network address of the data. To the user, all universes look alike; there seems to be no difference between reading or writing a file contained on a private disk, and reading or writing a file on a disk in the next building. The file is never copied over and worked on - all modifications to it are done over the network.

A user on one computer system could type in the following command to "mount" either a directory or a file system from another system on the network:

```
training$ mount -t nfs sn218:/trng/rjj/progs /u1/rjj
```

In the above example the user is on computer **training** and is asking the system to "connect" directory /trng/rjj/progs on **sn218** to the training file system under /u1/rjj. During this login session the user now would have directory /u1/rjj/progs available to use or modify.

NFS (Network File System)



X WINDOW SYSTEM

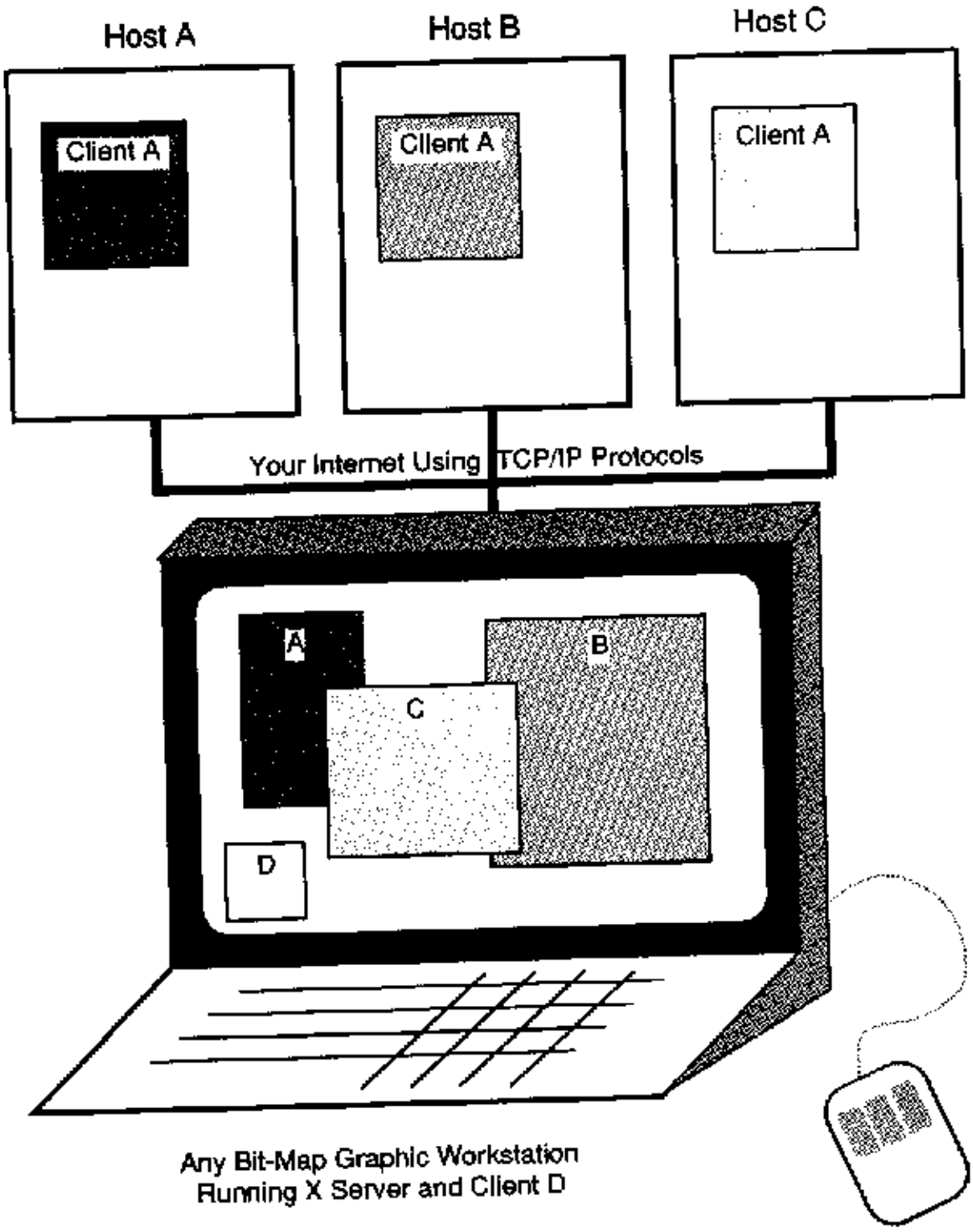
The X Window System, created at the Massachussetts Institute of Technology (MIT), was designed to provide multiple network sessions to be monitored on a computer terminal at the same time using different parts of the terminal screen. It is now in the public domain and is a product of many computer companies, including Cray Research.

X Windows is of great importance to Cray computer users because it allows supercomputer applications to produce graphic output without concern for the type of bit-mapped graphic workstation that will be used to display the results. Common bit-mapped graphic workstations that use X Windows include Sun, Apollo, and Silicon Graphics.

Most network applications utilize a client-server model of interaction. The server runs in the bit-mapped graphic workstation and the clients run either locally in the workstation (i.e., multiple sessions on the local computer) or remotely in another network host. The server is designed to work on that particular brand of workstation and knows how to display text and graphics.

It receives requests from clients saying things like "draw a rectangle with corners such-and-such and line thickness such-and such" and it issues the appropriate operating system instructions to perform this function on the hardware. Although handled differently by different servers, what shows up on the screen is the same - the rectangle is drawn on the screen.

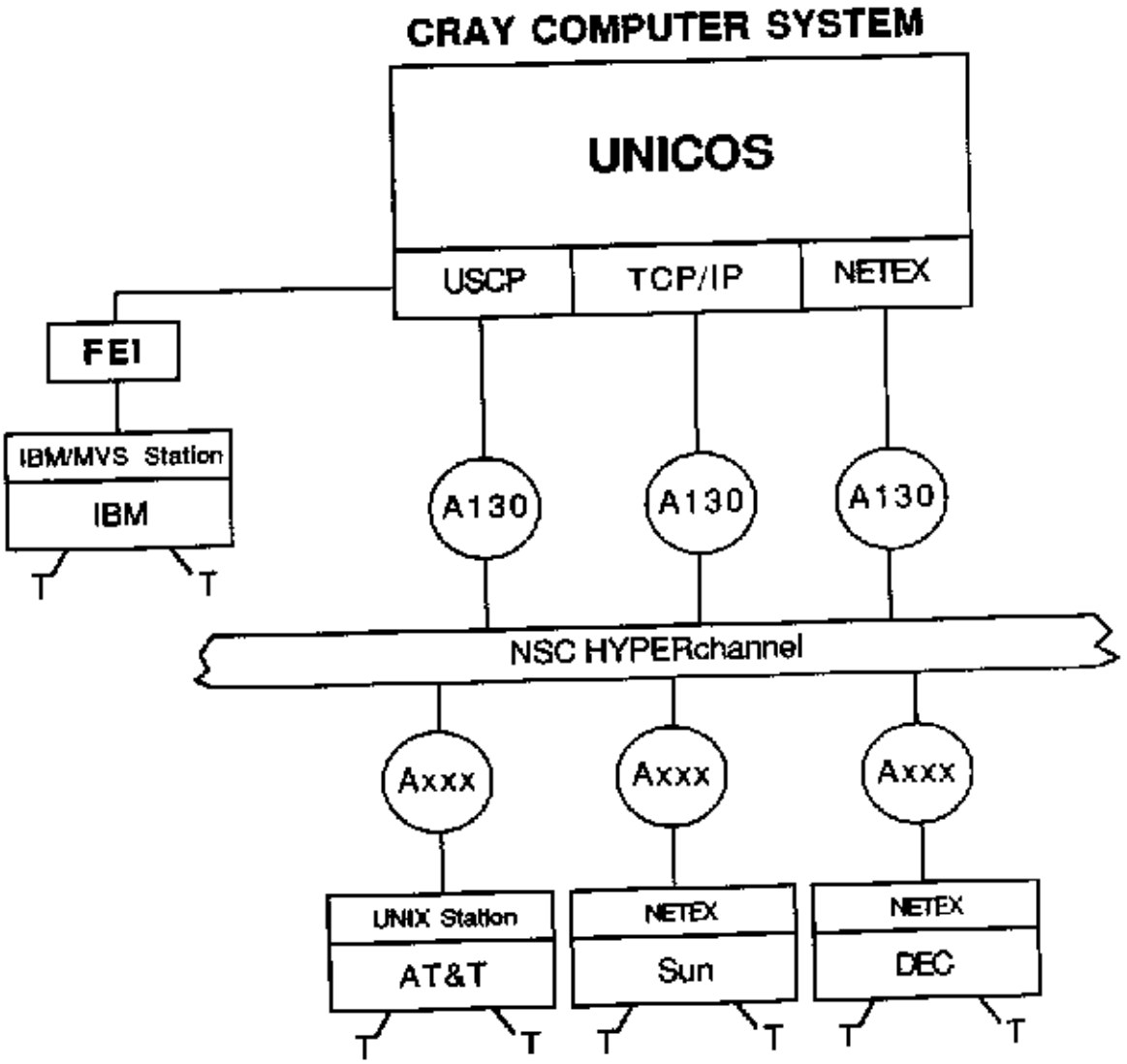
X Clients and Servers



NETEX

Netex is Network Systems Corporation's (NSC) proprietary network protocol which is designed specifically to run on a HYPERchannel network. While it has many TCP/IP-like functions, it is not as popular as TCP/IP.

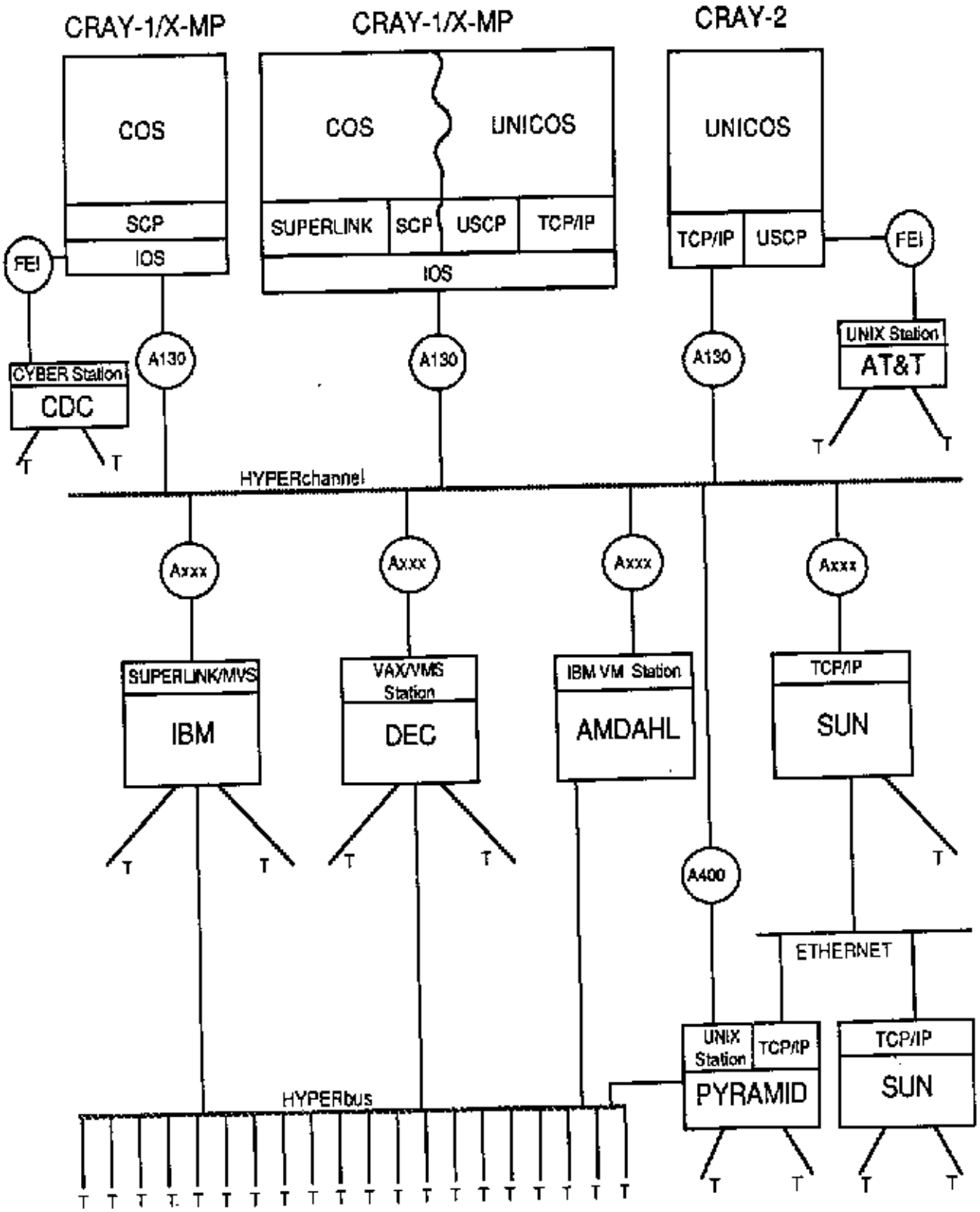
NETEX CONFIGURATION EXAMPLE



NETWORK INTEGRATION

The many different networking options available with Cray computer systems can be integrated into a large network which can accommodate a site's computing needs. Such a networking scheme allows users to share data between machines as well as send messages and monitor program execution.

CRAY NETWORKING SCHEMES



THIS PAGE IS INTENTIONALLY LEFT BLANK

SECTION 10

Cray Product Familiarization

CRAY PROGRAMMING LANGUAGES

Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK.

CRAY PROGRAMMING LANGUAGES

As fast and efficient as the hardware is on Cray computer systems, it needs to be told what to do and when to do it. This is done by using the many available programming languages and associated system programs.

There are currently six programming languages available for use by Cray customers plus one for Cray systems personnel to program the IOS hardware. Some of the languages are available in different forms for different machines. The languages fall into two categories: assemblers and compilers.

Assemblers

Cray Assembler Language (CAL, CAL Version 2)
A-Processor Middle Language (APML)

Compilers

C
Cray Fortran (CFT, CFT2) and Cray Fortran 77 (CFT77)
Pascal
Ada
Lisp

CRAY ASSEMBLER LANGUAGE (CAL)

CAL is the programming language available to Cray programmers who need to write programs at the individual hardware instruction level. CAL provides the user with a powerful symbolic language for generation of object code to be loaded and executed on the mainframe of a CRAY-1 or CRAY X-MP system. The COS Operating System is written in CAL as well as the original Cray Fortran compiler.

A new CAL assembler (CAL 2.0) supports all Cray systems including the CRAY-2 and future systems. It is written in Pascal.

CAL features include:

- Free field statement format
- Block control
- Powerful symbolic expressions
- Pseudo instructions
- Flexible data notation
- Powerful macro, micro, and opdef facilities
- Extensive set of useful macros provided through various libraries
- Listing control

CRAY ASSEMBER CODE EXAMPLE

```
**************************************************************
*                  This is a sample of CAL code              *
**************************************************************
```

```
      A7      64          Store the value 64 in A7
      VL      A7          Make the vector length 64
      AØ      VECTOR1     Put the address of VECTOR1 in AØ
      V1      ,AØ,1       Copy the 64 elements at  AØ to V1
      AØ      VECTOR2     Put the address of VECTOR2 in AØ
      V2      ,AØ,1       Copy the 64 elements at AØ to V2
      V3      V1+V2       Add V1 to V2 and store result in V3
      AØ      VECTOR3     Put the address of VECTOR3 in AØ
      ,AØ,1   V3          Copy the 64 elements in V3 to the
                          address stored in AØ
```

CRAY FORTRAN
(CFT, CFT2, CFT77)

Fortran (FORmula TRANslation) is the high-level programming language of choice among scientific programmers and because Cray computer systems are geared for handling the large-scale mathematical computations required by these programmers, Cray Research continues to offer improved versions of this compiler.

All versions of CRAY Fortran (**CFT, CFT2, CFT77**) include the following features:

- Full ANSI 78 (FORTRAN 77) implementation
- Cray Fortran language extensions
- Automatic vectorization
- Highly optimized scalar and vector code generation
- Powerful compiler directives
- Extensive scientific/arithmetic/utility subroutines provided through various libraries
- Macro- and microtasking capabilities

CFT is Cray's original Fortran compiler for code generation on Cray-1 and Cray X-MP systems. **CFT2** is Cray's version of the CFT compiler for code generation on CRAY-2 systems. Both compilers are written in CAL.

CFT77 is a new Fortran compiler implemented using the common modular compiling system (CMCS). It is written in the Pascal programming language which makes is more portable than CFT and CFT2. CFT77 generates optimized, vectorized code for all existing and future Cray systems. Its features include:

- Macro-, micro- and autotasking
- Automatic vectorization
- Support for existing CFT extensions
- Fortran 8X features
  - 31 character identifiers
  - Automatic arrays
  - Subset of the array syntax
- Future development:
  - Additional array syntax

CRAY FORTRAN CODE EXAMPLE

```
c*************************************************************************
c                        This is a sample of Fortran code                 *
c*************************************************************************

      subroutine mxm(a,msize1,b,msize2,c,msize3)

      dimension a(msize1,msize2),b(msize2,msize3),c(msize1,msize3)

      do 10 i=1,msize1
      do 10 j=1,msize3
        sum=0.0
        do 20 k=1,msize2
          sum=sum+a(i,k)*b(k,j)
   20 continue
        c(i,j)=sum
   10 continue
      return
      end
```

CF77

The **cf77** compiling system is a new way to easily invoke automatic parallelism in the compile process of CFT77 source code using only one command. It invokes preprocessors before it does the compile using CFT77 and produces executable output.

The processing phases include:

- **Dependency Analysis Phase**
  The Fortran Preprocessor (fpp) performs vectorization and autotasking dependency analysis. Directives are inserted into the code to pass this information to the other phases of the compiling system. Source code transformations are performed if applicable.

- **Translation Phase**
  The Fortran Mldprocessor (fmp) interprets directives supplied by fpp and/or user-supplied directives. Directives and their associated Fortran constructs are translated into a form that the compiler can use.

- **Code Translation Phase**
  The CFT77 compiler performs scalar and vector optimization and generates relocatable object code. If parallel constructs were detected in previous compiling system phases, code is generated to allow these parallel regions to execute on multiple processors.

The Fortran Compiling System is invoked using the cf77 command:

```
cf77 -Zp file.f
```

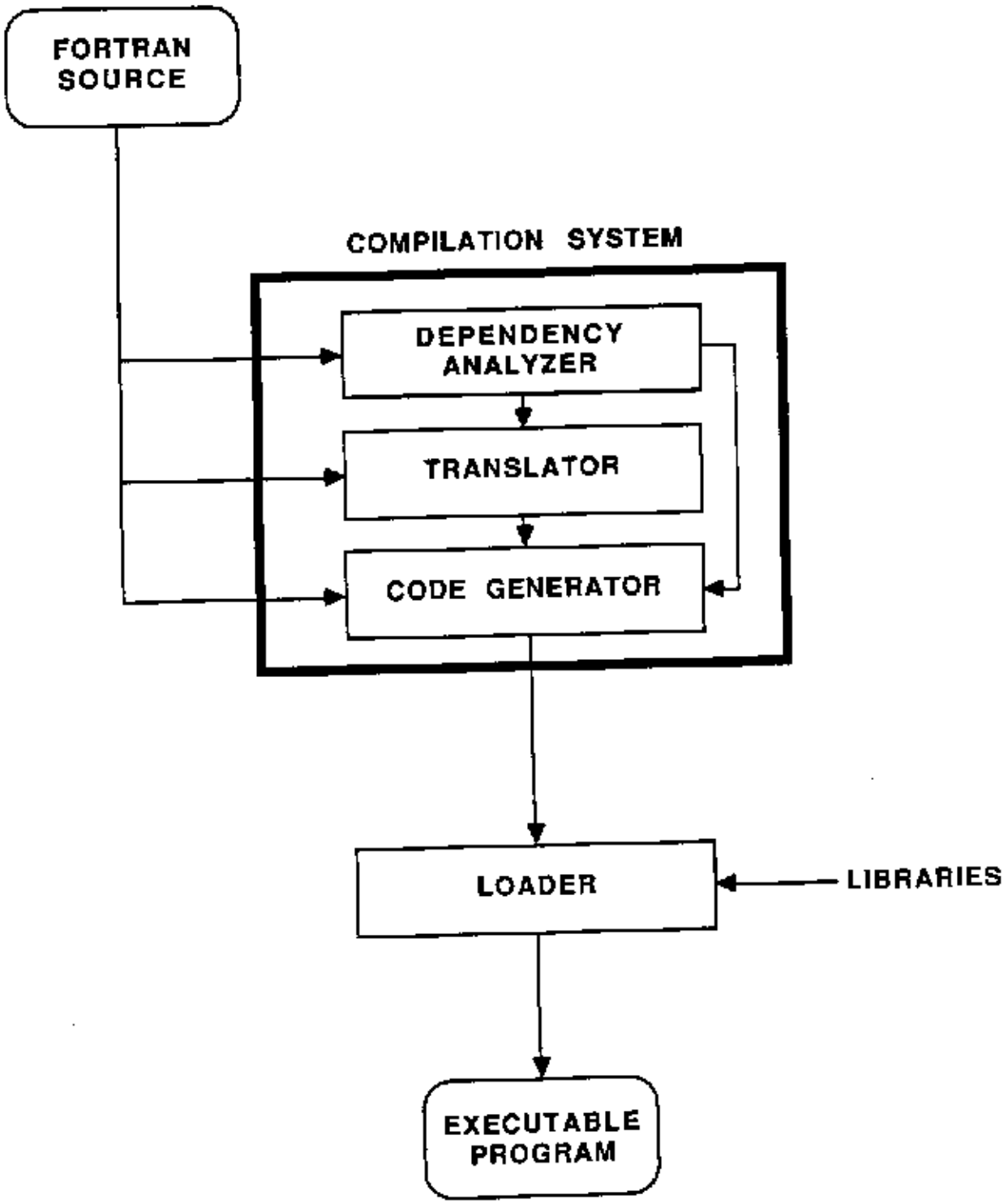
This command invokes the three compiling system phases described above (fpp, fmp, and code generation). It also links/loads object files and produces an executable binary output (a.out).

Below is an expanded version of what the above cf77 command generates:

```
fpp file.f > file.m
fmp file.m > file.j
cft77 file.j
rm file.m
rm file.j
segldr file.o
```

The Fortran Compiling System allows many other options on the control statement.

THE CRAY FORTRAN COMPILING SYSTEM



CRAY C COMPILER

The C programming language is a high-level language used extensively in the creation of the UNICOS operating system and the majority of the utility programs that make up UNICOS. The Cray C compiler runs under UNICOS and COS. It was developed by Cray Research from an earlier AT&T compiler written at Bell Labs for their Cray computer system.

C is a modern computer language that is available on processors ranging from microcomputers to supercomputers. C is useful as a language for a wide range of applications, especially test and system-oriented programs because of its low-level features. The availability of C nicely complements the scientific orientation of Fortran. It performs scalar optimization and vectorizes code automatically.

Cray C will soon be implemented using the Common Modular Compiling System (CMCS).

CRAY C CODE EXAMPLE

```
/*****************************************************************/
/*                        This is a sample of C code             */
/*****************************************************************/

#define MSIZE1 10
#define MSIZE2 10
#define MSIZE3 10

MXM(a,msize1,b,msize2,c,msize3)

   float a[MSIZE2][MSIZE1],b[MSIZE3][MSIZE2],c[MSIZE3][MSIZE1];
   int *msize1,*msize2,*msize3;
{
   float sum;
   int i,j,k;

   for(i=0; i<*msize1; i++) {
      for(j=0; j<*msize3; j++) {
         sum = 0.0;
         for(k=0; k<*msize2; k++) {
            sum = sum + a[k][i] * b[j][k];
         }
      c[j][i] = sum;
      }
   }
   return(0);
}
```

CRAY PASCAL COMPILER

Pascal is a modern, high-level, general-purpose programming language. Cray Pascal runs on all Cray machines and supports the International Standards Organization (ISO) Level-1 standard and ANSI Pascal standards with very minor exceptions. It offers such extensions to the standard as separate compilation of modules, imported and exported variables, and an array syntax.

The optimizing Cray Pascal compiler takes advantage of Cray hardware features through both scalar optimization and automatic vectorization of FOR loops. It provides access to Fortran common block variables and uses a common calling sequence that allows Pascal code to call Fortran, C, and CAL routines.

Cray Pascal provides excellent debugging facilities and is used as a system implementation language for the CMCS.

CRAY PASCAL CODE EXAMPLE

```
(*************************************************************)
(*                  This is a sample of Pascal code          *)
(*************************************************************)


MODULE MXMP;

CONST MSIZE1 = 10;
    MSIZE2 = 10;
    MSIZE3 = 10;
TYPE  MATRIXA = ARRAY [1..MSIZE2,1..MSIZE1] OF REAL;
    MATRIXB = ARRAY [1..MSIZE3,1..MSIZE2] OF REAL;
    MATRIXC = ARRAY [1..MSIZE3,1..MSIZE1] OF REAL;
    INT = INTEGER;

PROCEDURE MXM (A:MATRIXA; MS1:INT; B:MATRIXB; MS2:INT;
                VAR C:MATRIXC; MS3:INT); EXPORTED;

VAR  I,J,K : INT;
   SUM  : REAL;

BEGIN
FOR I := 1 TO MS1 DO
  FOR J := 1 TO MS3 DO
    BEGIN
    SUM := 0;
    FOR K := 1 TO MS2 DO
      SUM := SUM + A[K,I] * B[J,K];
    C[J,I] := SUM;
    END;
END;
```

A-PROCESSOR MIDDLE LANGUAGE (APML)

APML is the programming language used to program the I/O Subsystem operating system. It combines the characteristics of a macro assembler and a compiler.

APML is used only by Cray software developers and system analysts.

CRAY APML CODE EXAMPLE

```
**************************************************************************
*                          This is a sample of APML code                 *
**************************************************************************

        GET  S7,ETL,SB          .time limit (lower)
        $IF (S1 = 0)
          CALL BTD,(S7,0,24,LB,S3,1,S2) .output lower bits value
        $ELSEIF (S1 = 1)        .time greater than D'65535
         $IF (S7 <= D'34463)    .65536 + 34463 = 99999
GET2        *
         GETMEM   4,R!BUF       .place to store both parcels
         $IF (A#0)              .buffer not available
          PAUSE 1               .wait 1/10 second
          P=GET2                .try again
         $ENDIF
         CLEAR  START=R!BUF,COUNT=4
         (R!BUF) = S1
         (R!BUF + 1) = S7       .put parcels in one spot
          CALL  BTD,(R!BUF,14,18,LB,S3,1,S2) .binary to decimal
         RELMEM  R!BUF
        $ELSE
         S1 = R!%B + MXTM
         R = MOVEIT
        $ENDIF
       $ELSE                    .time too big, print *****
        S1 = R!%B + MXTM
        R = MOVEIT
       $ENDIF
```

CRAY ADA COMPILER

In 1975, the U.S. Department of Defense (DOD), seeking an alternative to maintaining hundreds of specialized computer languages used in strategic applications, put out a call for a single new language that could handle its diverse needs. The result was Ada, which incorporates many modern software development principles.

Ada is a new addition to the list of available Cray compilers. It is a language well-suited for large, complex software systems, and was designed to help scientific and engineering users develop large, real-time embedded systems. Application areas include process and design control, communications systems, intelligent systems, geophysical analysis, robotics, distributed applications, and management information systems.

Because of its powerful capabilities it is necessarily also a very large compiler and is slower to compile and execute than languages like Fortran. But it also has many features that are unavailable in Fortran including things like exception handling, low-level I/O, and abstract data types.

Ada is especially well suited for developing large, complex software systems. For example, it encourages separate compilation and testing of packages that will be combined later.

Release 1.0 of Cray Ada runs on CRAY X-MP and CRAY-2 systems under UNICOS 4.0 or later. Ada Release 2.0 will also run on CRAY Y-MP systems.

Cray Ada comforms to the Ada language as defined in the American National Standard Reference Manual for the Ada Programming Language (LRM), the official authoritative source on the syntax and semantics of the Ada language. It also passes the Ada Compiler Validation Capability (ACVC) test suite, required by the DOD.

CRAY ADA CODE EXAMPLE

```
--  ************************************************************
--  *                    This is a sample of Ada code          *
--  ************************************************************

FUNCTION "*" (Left,Right:Matrix_Type) RETURN  Matrix_Type IS
      Result : Matrix_Type (Left'Range(1),Left'Range(2));
BEGIN
      FOR i IN Result'Range(1) LOOP
            FOR j IN Result'Range(2) LOOP
               Result (i,j) := Left (i,j) * Right (i,j);
            END LOOP;
      END LOOP;
      RETURN Result;
END "*";
```

CRAY LISP COMPILER

Artificial intelligence (AI) has long held the promise of a computer system that can think for itself and learn from its mistakes. Lisp, the predominant language of AI, brings this goal a step nearer to being realized. However, the tools and techniques of AI, such as rule-based expert systems and hierarchically structured tree searches, require extensive memory and vast numbers of logical operations to perform meaningful work in a practical time frame. Cray computer systems, with their very large memories and fast processors, can overcome these obstacles when solving real-world AI problems.

Cray Allegro Common Lisp is industry-standard Common Lisp with extensions which take advantage of the power of Cray systems. The initial release of Allegro Common Lisp for Cray computer systems runs under UNICOS 4.0 or later on CRAY X-MP-based systems.

Lisp (which stands for list processing) is quite different from other programming languages. In Lisp, program behavior can be modified during execution as new information is obtained, providing a very effective problem-solving tool. Lisp programs are built the way a human knowledge base is built: incrementally, using and building on what is already known.

Applications which are well-suited to Cray Allegro Lisp include artificial intelligence; natural language processing (including speech recognition); image processing; automatic theorum proving; rule-based expert systems that eliminate human experts in fields such as aerospace, geology, and automotive design; software prototyping and automatic programming; diagnostics and "what if?" scenarios.

Since many other computers run a version of Lisp, programs written on them can easily be ported to Cray systems.

CRAY LISP CODE EXAMPLE

```
;*****************************************************************
;*                    This is a sample of Lisp code              *
;*****************************************************************

(RULE IDENTIFY6
      (IF (ANIMAL HAS POINTED TEETH)
            (ANIMAL HAS CLAWS)
            (ANIMAL HAS FORWARD EYES))
      (THEN (ANIMAL IS CARNIVORE)))


(DEFUN FIBONACCI (N)
      (COND ((= N 0) 1)
                  ((= N 1) 1)
                  (T (+ (FIBONACCI (- N 1))
                        (FIBONACCI (- N 2))))))
```

COMMON MODULAR COMPILING SYSTEM (CMCS)

The Common Modular Compiling System (CMCS) at Cray Research is a new means of creating high-level language compilers. A program broken down by a compiler created by the CMCS passes through three independent phases. Those phases are:

- Phase I - Language specific syntax analysis
  This phase analyzes the language-specific syntax (e.g., Fortran code statements).

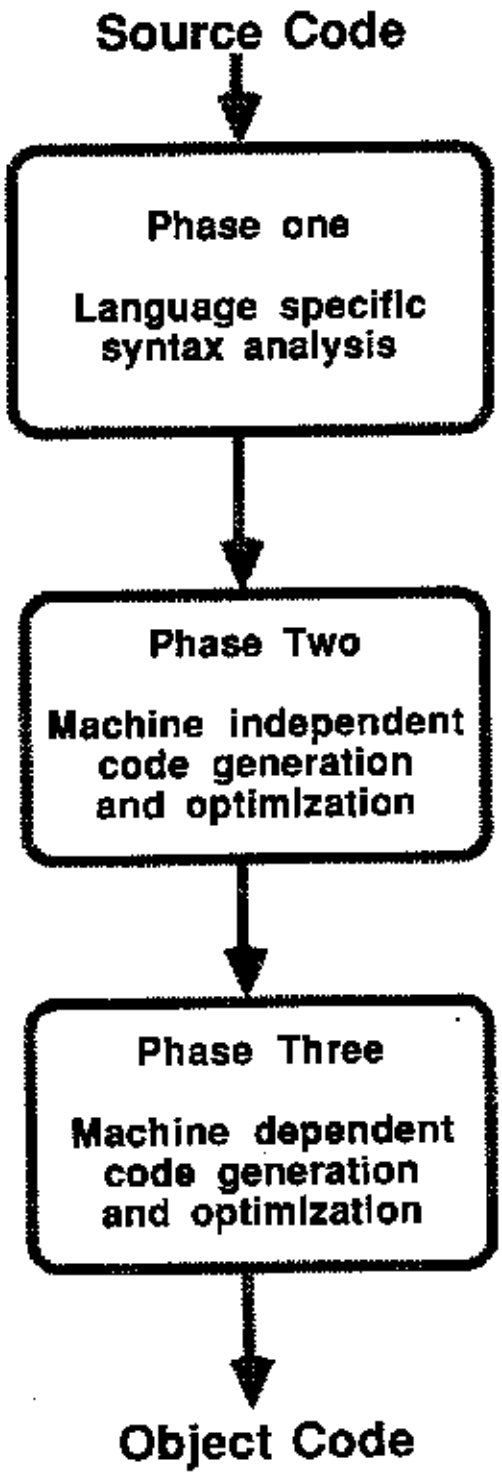
- Phase II - Machine independent code generation and optimization
  This phase optimizes the program to run more efficiently on any Cray mainframe (machine independent). This includes vectorization, parallelism, and elimination of redundant code and "dead code".

- Phase III - Machine dependent code generation and optimization
  This phase optimizes the program to run more efficiently on a specific Cray mainframe (e.g., Cray-2 or Cray X-MP). The instructions for the specific machine are created in this phase.

The importance of this whole concept is that the creation of a new compiler for a different language or a new Cray computer does not require rewriting the whole compiler, only the phase affected by the change. For example, CFT77 was written using the CMCS. A C compiler written with the CMCS will soon be available. For that compiler creation effort only Phase I needs modification. Future machines (new hardware) will require only the modification of Phase III for each language.

Other language compilers written with the CMCS will be available in the future.

Common Modular Compiling System



PROGRAM FLOW AND SYSTEM LIBRARIES

A source program must be compiled or assembled before it can execute as a program in a CPU. The primary programming languages offered by Cray Research include CAL (Cray Assembler Language), CFT77 (Cray Fortran), C, and Pascal.

The assembly process (for CAL) or the compile process (for the other languages) deciphers the particular language statements into binary (ones and zeroes that the CPU understands). There are, however, many commonly used routines programmers call for in programs so they don't need to write them themselves. This code needs to be plugged into the program at the appropriate location. These routines are already assembled/compiled and are stored in **binary libraries**. The assembler/compiler flags those routines so the next step (**SEGLDR**) can find them.

SEGLDR's job is to take the deciphered binary code (SEGLDR does not know or care what programming language the source was written in) and locate the flagged routines the assemble/compile process flagged. These are available in binary libraries at SEGLDR's disposal. Some of the main libraries include (COS names start with *$* and UNICOS names start with *lib*).

$ARLIB (libm.a) contains routines primarily concerned with returning some numeric result. Examples are sine, cosine, and square root.

$FTLIB (libf.a) contains CFT-specific routines such as formatted I/O routines.

$IOLIB (libio.a) contains routines that move data from external devices to main memory or that controls that movement such as translating IBM data format (EBCDIC) to CRAY data format (ASCII).

$SCILIB (libsci.a) contains routines that perform operations such as matrix multiply. The routines must be explicitly called because they are not built-in routines in the programming languages (such as CFT77) and are not part of the operating system.

$UTLIB (libu.a) contains routines which are of a utilitarian nature such as managing memory, getting the system date and time, and sorting/merging data.

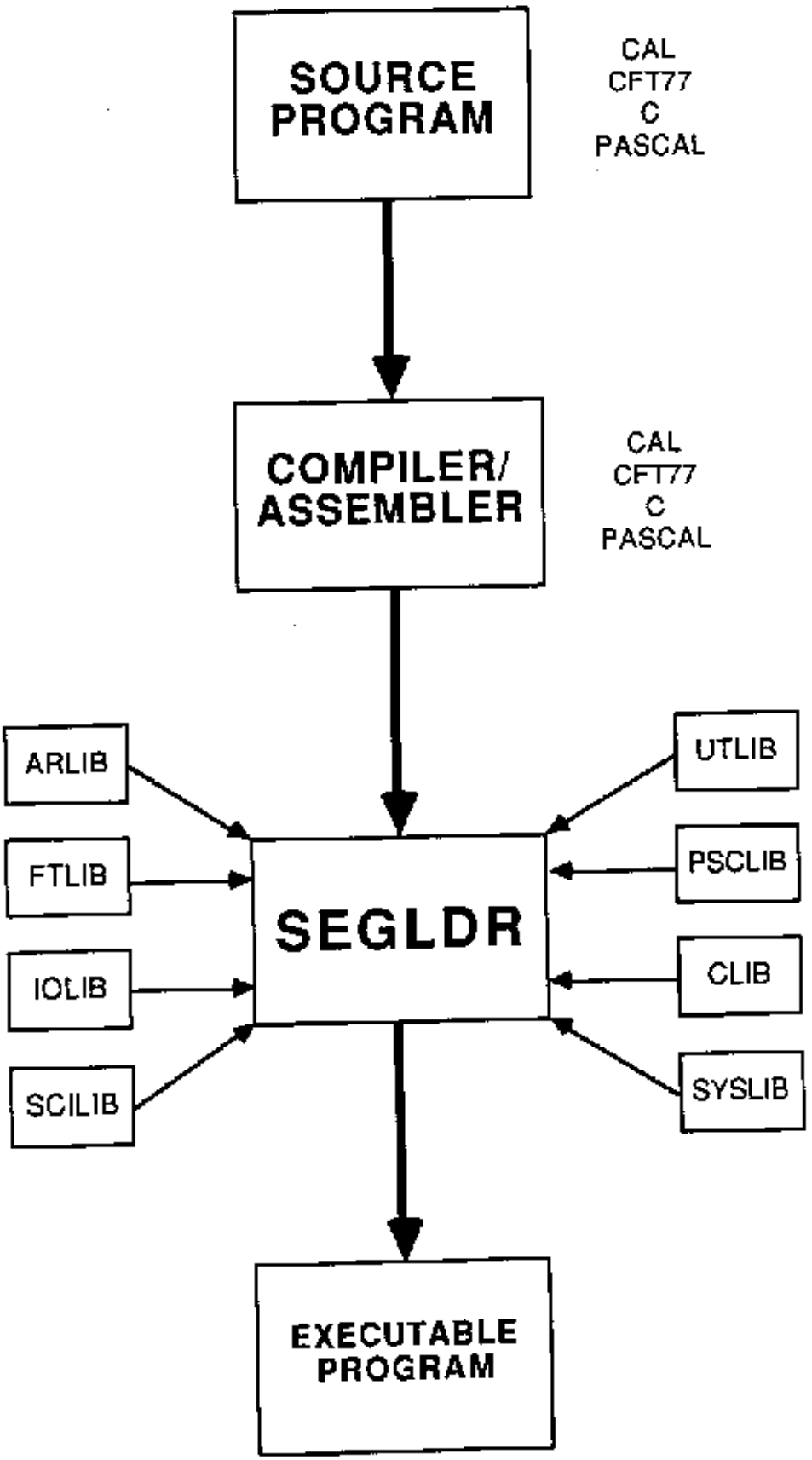
$PSCLIB (libp.a) contains Pascal-specific routines such as formatted I/O routines.

$CLIB (libc.a) contains C-specific routines such as formatted I/O routines.

$SYSLIB contains routines which link a program directly to the COS operating system. In general, $SYSLIB serves as a link between general purpose $IOLIB and $UTLIB routines and COS.

Once all the required routines are located and inserted into the "bit stream" of the program, it is ready to execute.

PROGRAM GENERATION



THIS PAGE IS INTENTIONALLY LEFT BLANK.

SECTION 11

Cray Product Familiarization

CRAY SYSTEM PERFORMANCE

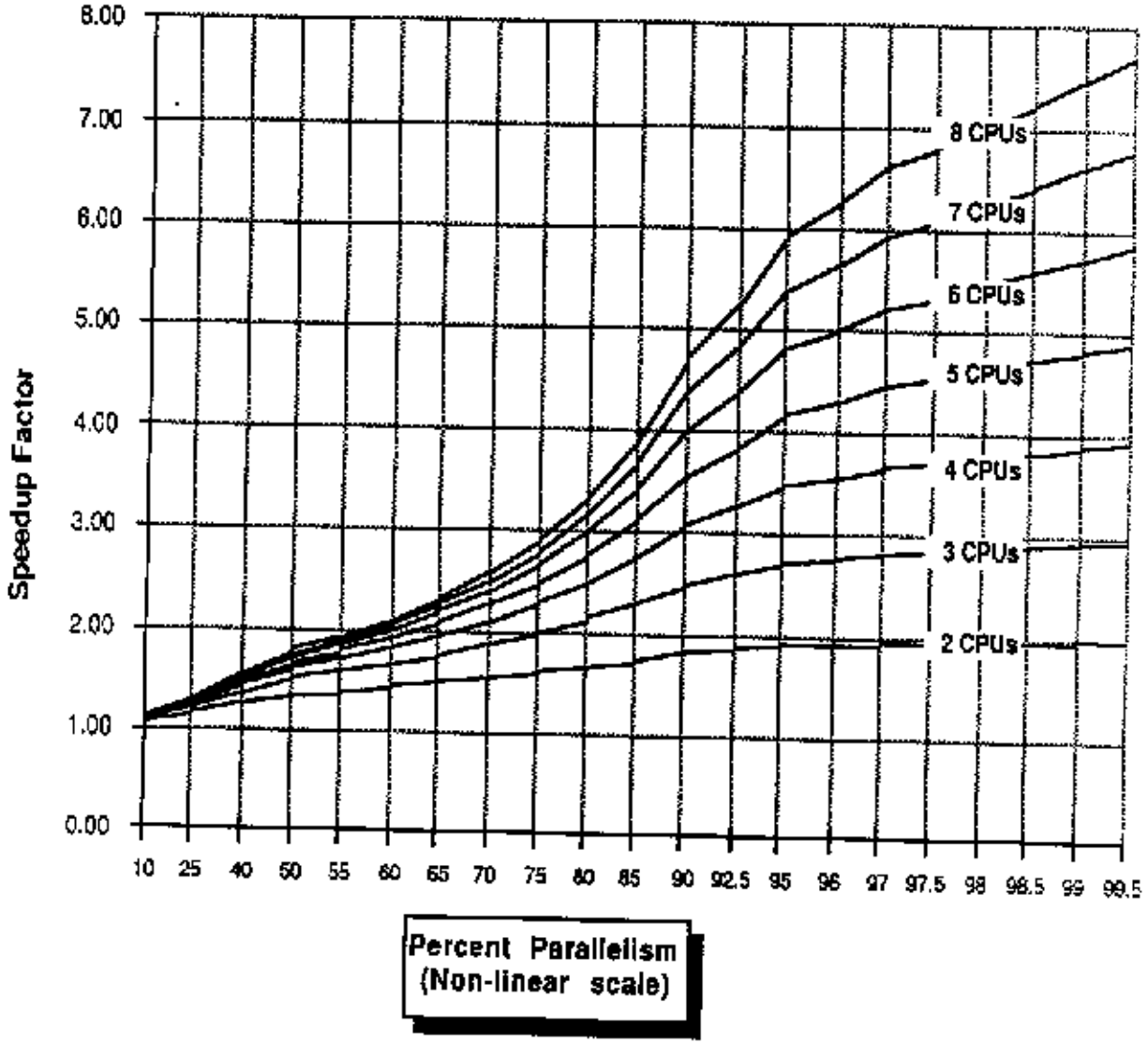
Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK.

CRAY PERFORMANCE COMPARISONS

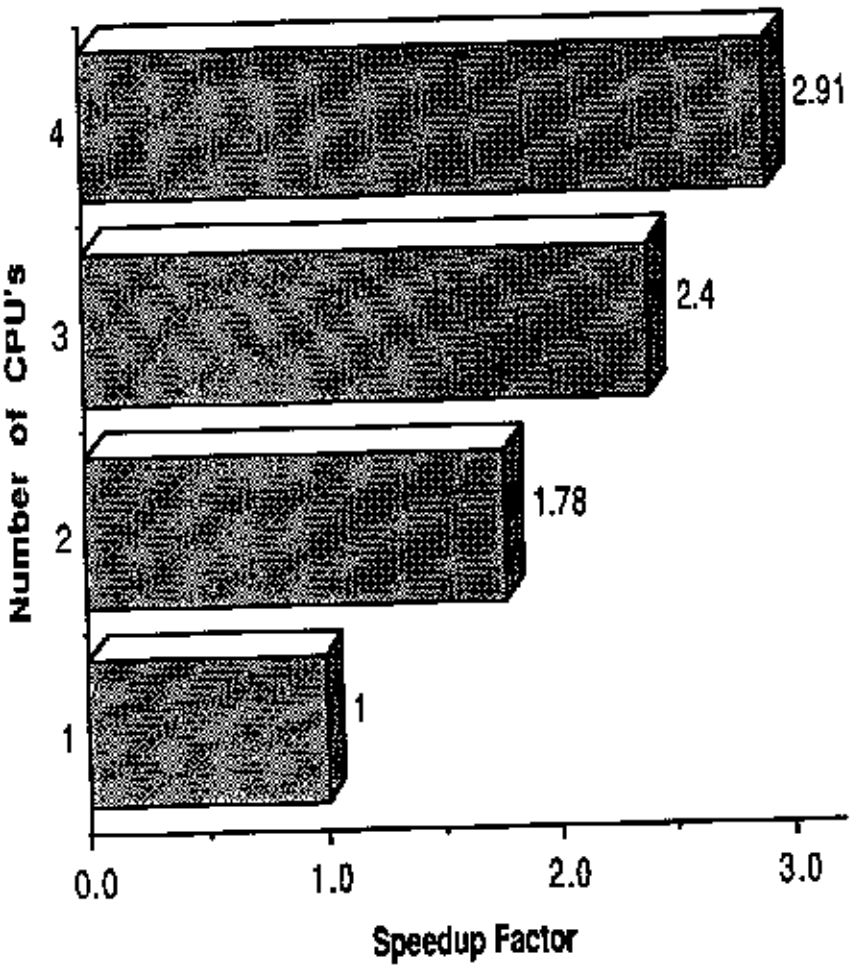
CRAY-1A	1.0
CRAY X-MP/1	1.5-2.5
CRAY X-MP/2	3-5
CRAY X-MP/4	6-10
CRAY-2	6-12
Y-MP	20-30

Amdahl's Law for 2 to 8 CPUs

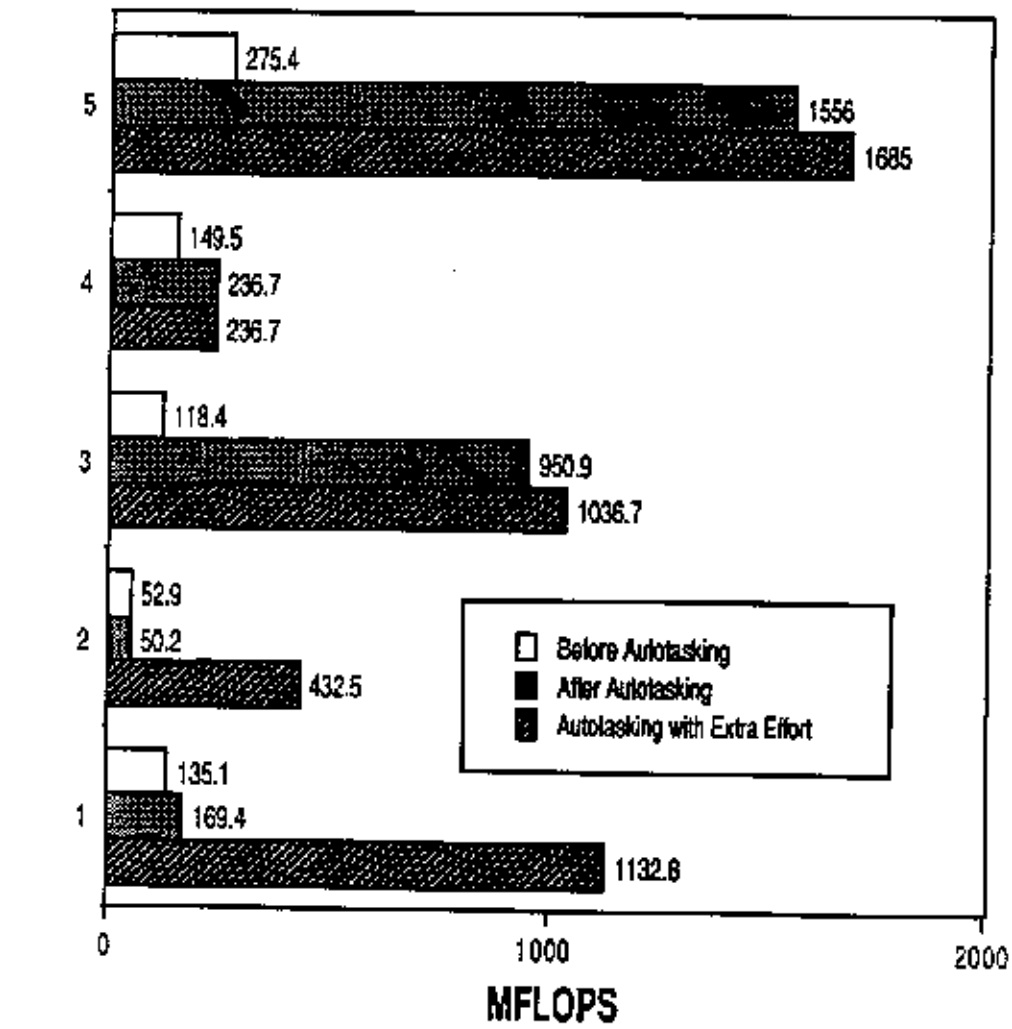


Autotasking Results

A Parallel Magnetohydrodynamics Code

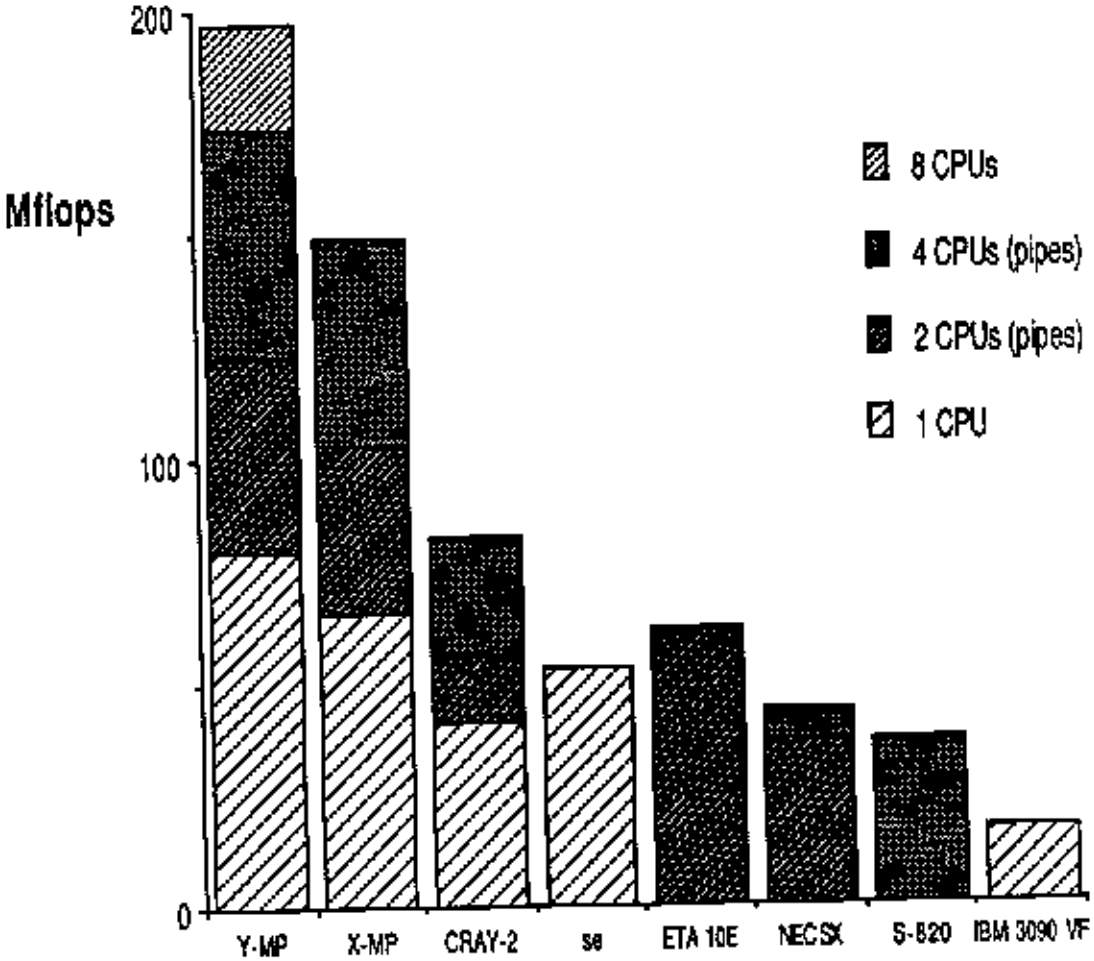


Autotasking Results

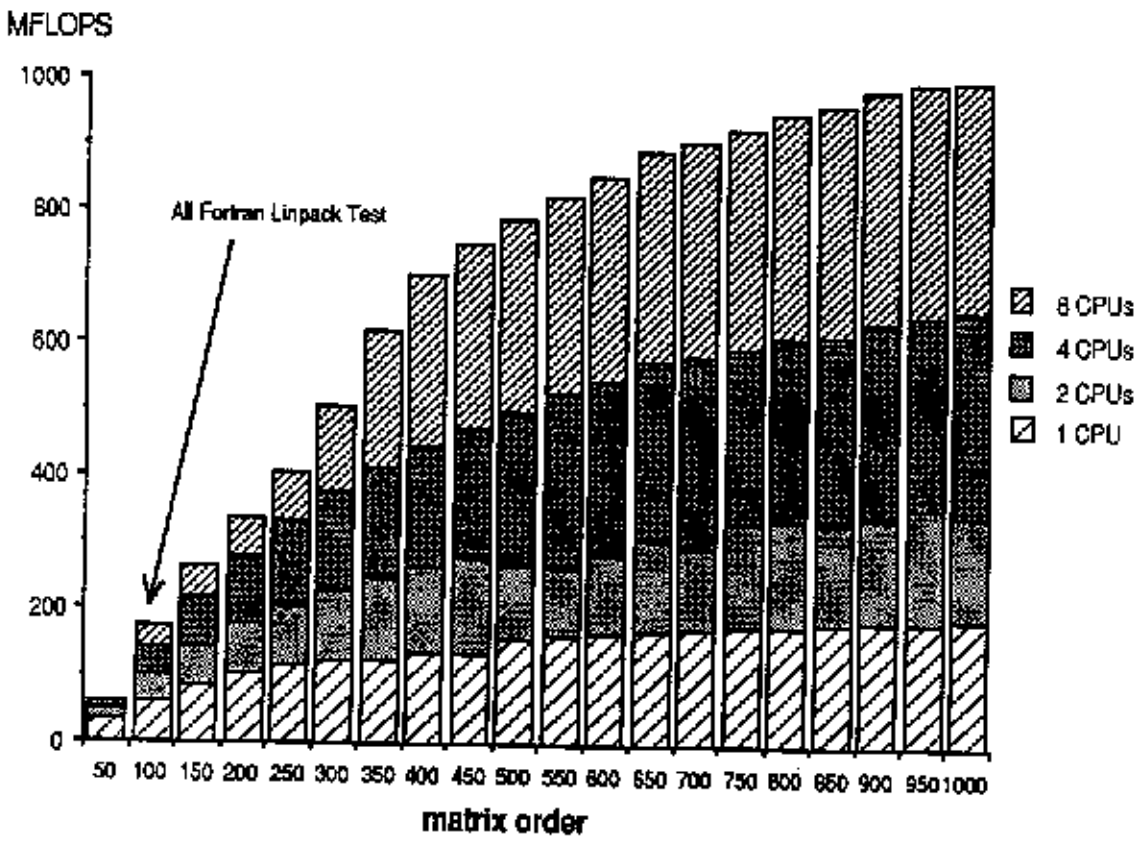


Autotasking - Linpack Test

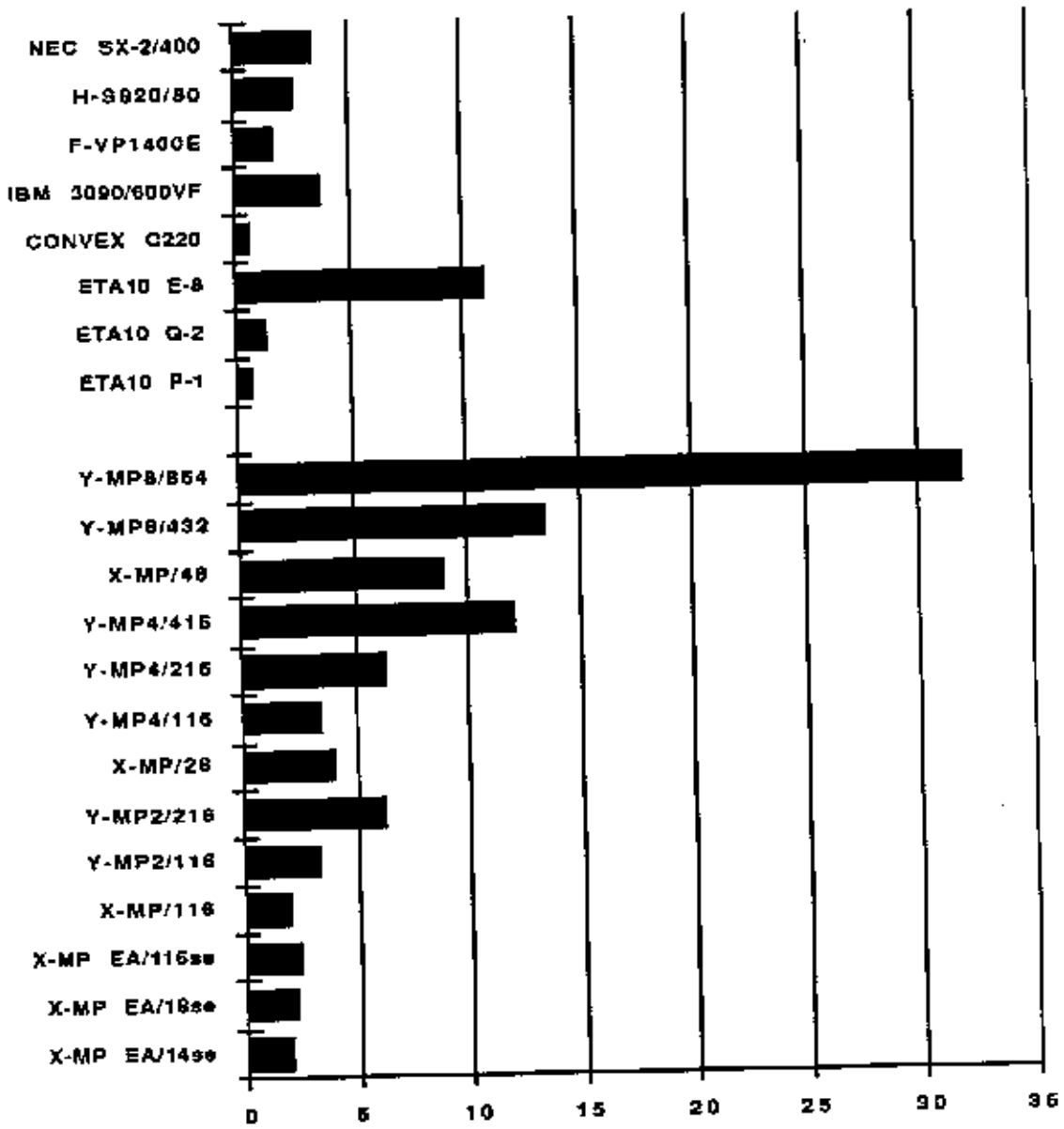
LINPACK TEST: ALL FORTRAN



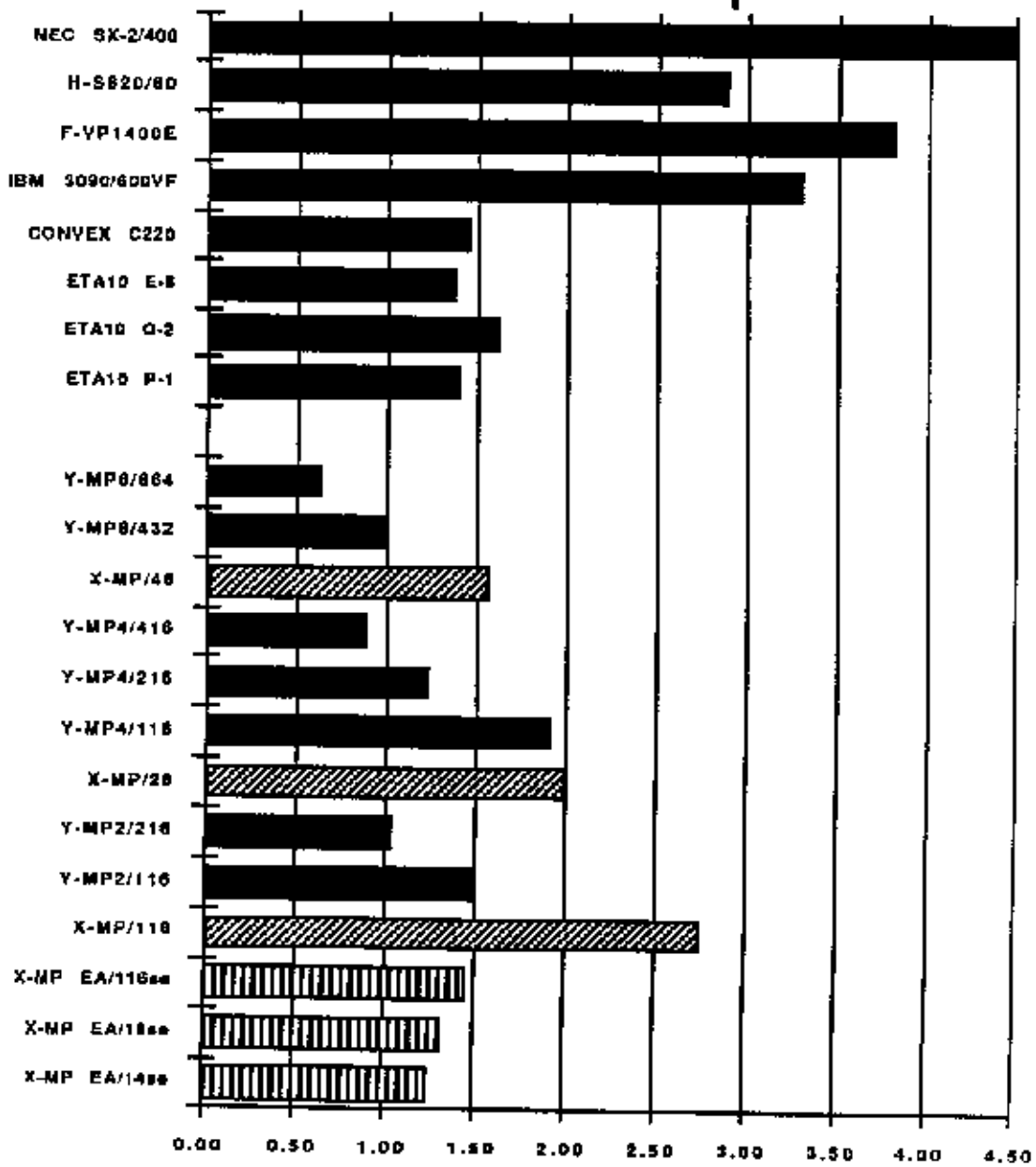
CRAY Y-MP8 Autotasking



Performance Times a CRAY-1



Price/Performance Comparison



CRAY Y-MP8 Hero Problem

PROBLEM:

- **COMPLEX LU DECOMPOSITION**

  40,000 X 40,000

- **OPERATIONS = $8/3n^3$ = 170 TRILLION (10^{12}) OPS**
- **STORAGE = 40,000 * 40,000 * 2 = 3.2 BILLION WORDS**

RESULTS:

- **PERFORMED ON S/N 1010, Y-MP8/864 + 128MWORD SSD**
- **16.8 HOURS ELAPSED TIME, 30 MINUTES I/O WAIT TIME**
- **2.1 GIGAFLOPS SUSTAINED ON 8 CPUS**
- **2000 BILLION (10^9) BYTES OF I/O**
- **CRAY WAS THE ONLY VENDOR ABLE TO DO THE PROBLEM**

Y-MP8 Gigaflop Performance

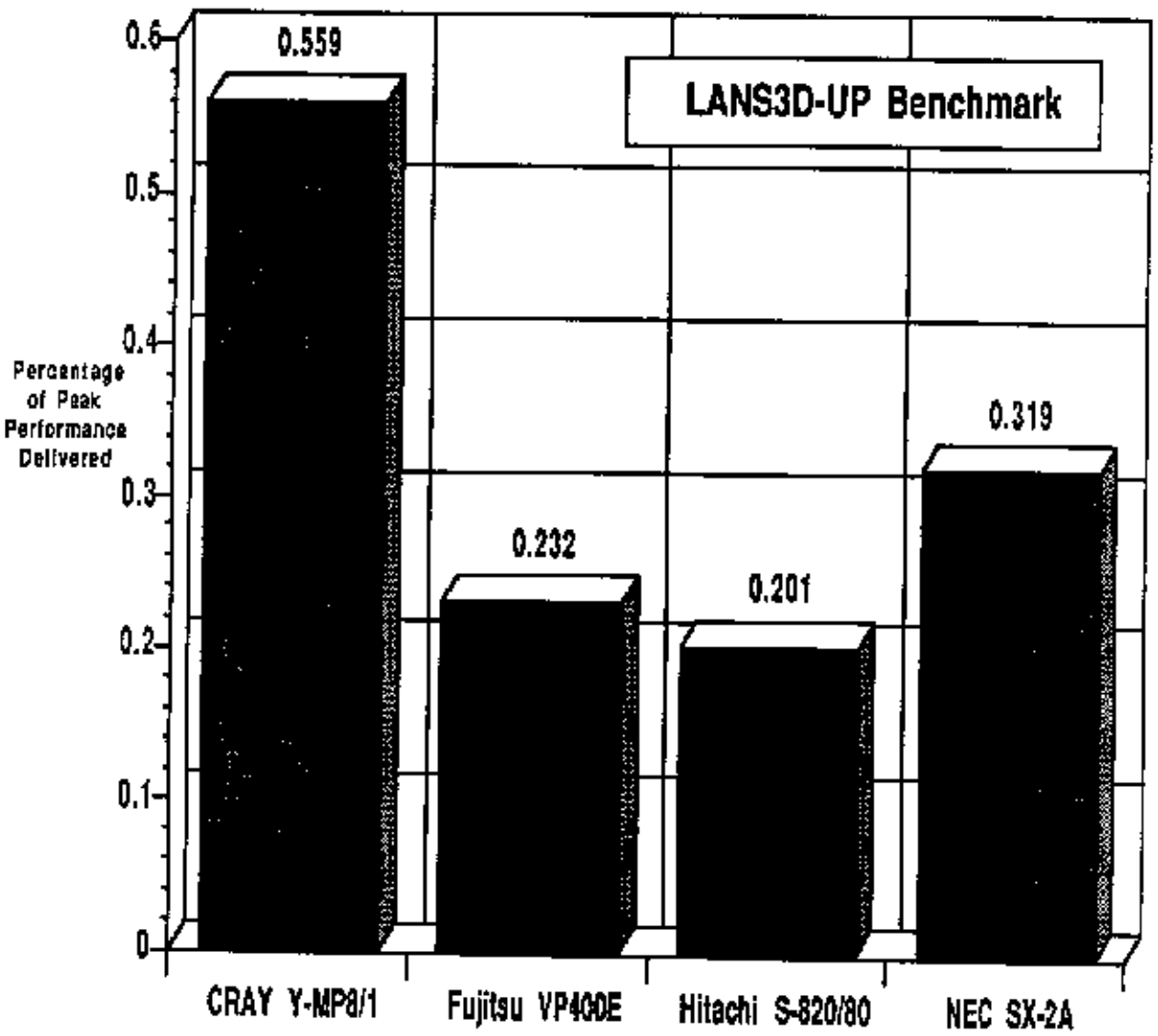
Institution	Program	CPU Secs.	GFLOPS
NASA Langley	Space shuttle - SRB	6.82	1.343
NASA Ames	2D Navier-Stokes	1444.8	1.084
NCAR	Shallow Water Model	14.77	1.326
U. of Tel-Aviv	3D post-stack migration	124.1	1.289
Exxon R&D	Elastic wave model	33.44	1.266
Cray Research	Image Reconstruction	57.782	1.068
Cray Research	Molecular structure	11.0	1.125
Cray Research	Complex LU decomposition	299395	2.116

Institute for Space & Astronautical Science
Kanagawa, Japan

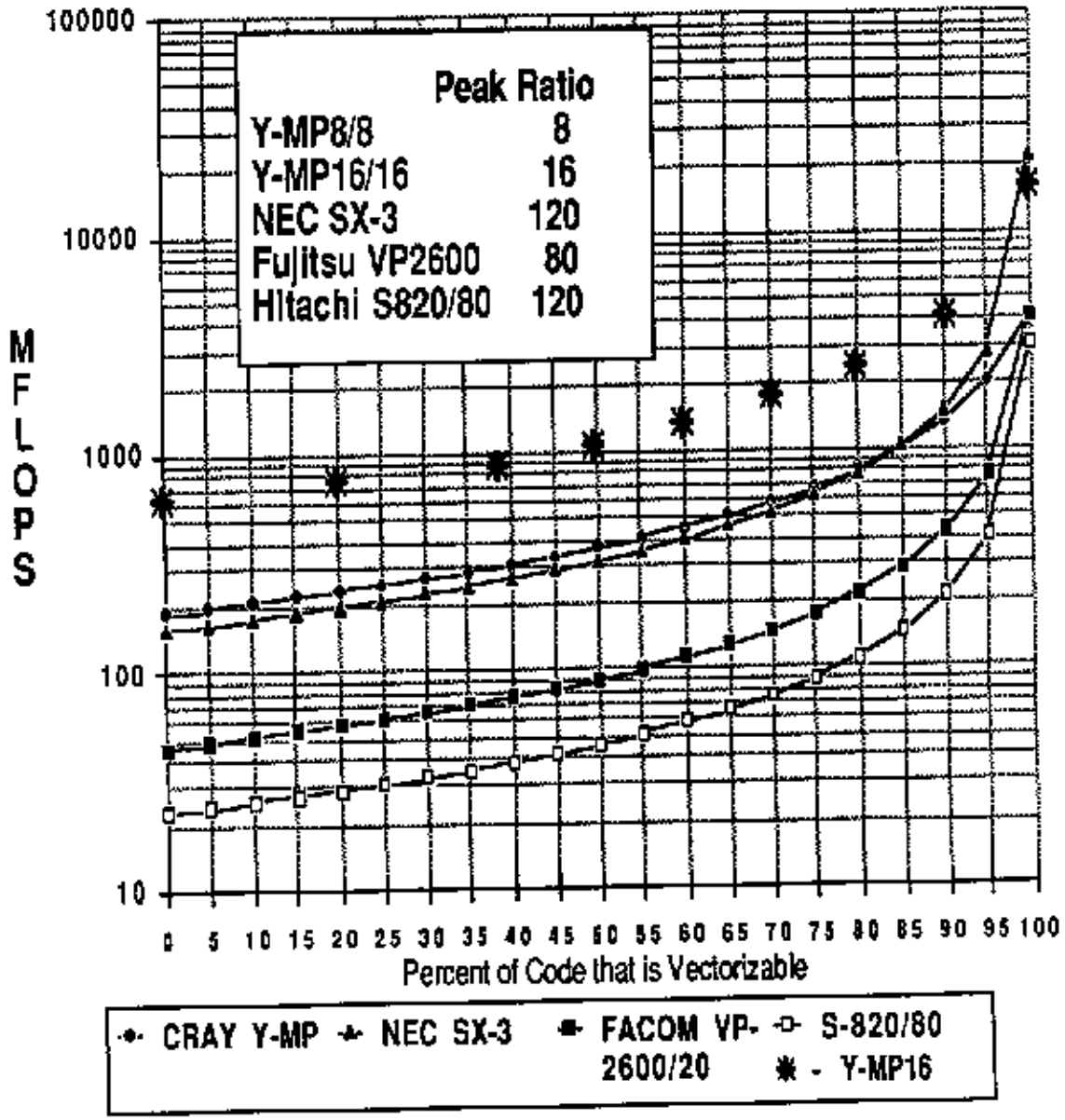
LANS3D-UP - Navier Stokes Code by K. Fujii and Y. Tamura, 1989
Subsonic, turbulent, high angle-of-attack airflow
200 Iterations, 853,349 mesh points, 26 MWords of memory

Computer System (Single Processor)	Peak Speed (GFLOPS)	Actual Speed (GFLOPS)	Actual/Peak Ratio
CRAY Y-MP8/832	0.334	0.186	0.559
Fujitsu VP-400E	1.7	0.395	0.232
Hitachi S-820/80	3.0	0.602	0.201
NEC SX-2A	1.3	0.414	0.319

Institute for Space & Astronautical Science Kanagawa, Japan



Supercomputer Performance Analysis



THIS PAGE IS INTENTIONALLY LEFT BLANK.

SECTION 12

Cray Product Familiarization

CRAY APPLICATIONS

Cray Research, Inc. Software

THIS PAGE IS INTENTIONALLY LEFT BLANK.

Supercomputer applications

Cray Research computer systems generally are acknowledged to be the world's most powerful general-purpose computers. Their rapid processing speeds and large memories have enabled scientists and engineers to conduct research and solve problems in a novel way. Along with the traditional methods of science — developing theories and testing them with physical experiments — researchers use CRAY systems to model physical structures and processes mathematically.

Computer models are valuable tools for studying phenomena too time-consuming, dangerous, expensive, or difficult to study experimentally. Although conventional computers can be used for modeling, real-world problems are often too large to make modeling on conventional computers practical. Days, weeks, or even months can be required to run realistic models on conventional computers because of their slow processing speeds and small memories.

But CRAY systems provide the rapid processing and massive storage capacities that researchers need to run realistic models in a practical timeframe. Large three-dimensional models with complex geometries and many interacting variables can be run cost-effectively on CRAY systems. Their extraordinary processing capacity has established CRAY computer systems as the standard for state-of-the-art research, providing industry, education, and goverment with accurate, detailed, and profitable results.

Aerodynamic simulation

Airplane designers have long relied on wind tunnels to evaluate the aerodynamics of airplanes and airplane sections. But wind tunnel tests require the time-consuming and costly construction of physical test models. Wind tunnels are also expensive to maintain and cannot detect certain air flow

phenomena. CRAY systems enable airplane designers to bypass many cycles of wind tunnel testing. Computational modeling on CRAY systems enables designers to test and modify designs quickly and cost-effectively, while conserving resources that would otherwise be spent testing physical prototypes. In addition, the ease with which designs can be modified on a computer gives designers freedom to experiment and encourages innovation. The auto industry is beginning to enjoy the benefits of aerodynamic testing via supercomputer and aerodynamic modeling is increasingly being used in the internal design of aircraft engines.

Seismic data processing

Many oil companies and geophysical contractors rely on CRAY systems to help locate petroleum deposits. CRAY systems are used to process the large volumes of data generated by reflection seismology, a technique used in geological exploration. Reflection seismology involves inducing a shock into the ground, usually with specially designed hydraulic devices, and recording the sound waves reflected back to the surface. Analyzing the recorded data can reveal important geological features that may indicate the presence or absence of petroleum. However, the amount of data needed to profile a large volume of earth accurately can be immense, and the required analyses are staggeringly complex. Only large-scale computers such as CRAY systems can perform the detailed analyses on these large data sets in a timely and cost-effective way. Seismic data processing on CRAY systems minimizes the liklihood of drilling unproductive wells, preserving the resources of petroleum companies and the environment.

Petroleum reservoir simulation

A number of major oil companies use CRAY systems to simulate the complex flow phenomena that characterize petroleum recovery operations. Such operations typically involve injecting water or water plus a surfactant into a reservoir

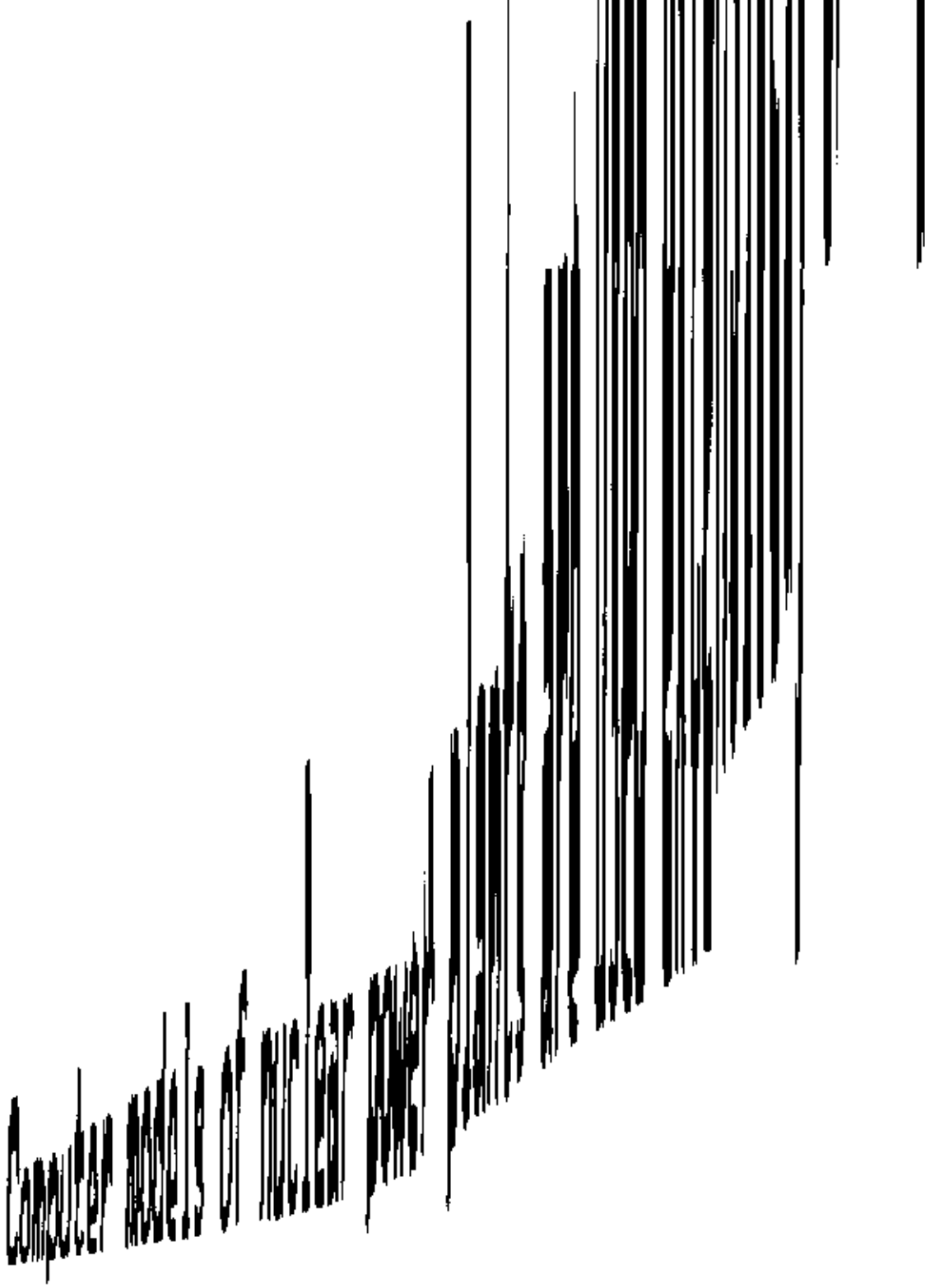
to mobilize the trapped oil and push it to the surface. However, to determine the best recovery strategy for a particular reservoir, engineers must consider the underground temperature and pressure, the chemical makeup of the petroleum, and the reservoir's geology. Modeling the complex interactions of these variables for large volumes of earth requires the kind of massive computing power CRAY systems provide. Because an increase in reservoir output of only a few percent can translate into millions of dollars, CRAY systems are among the petroleum industry's most valuable and cost-effective research tools.

Structural analysis

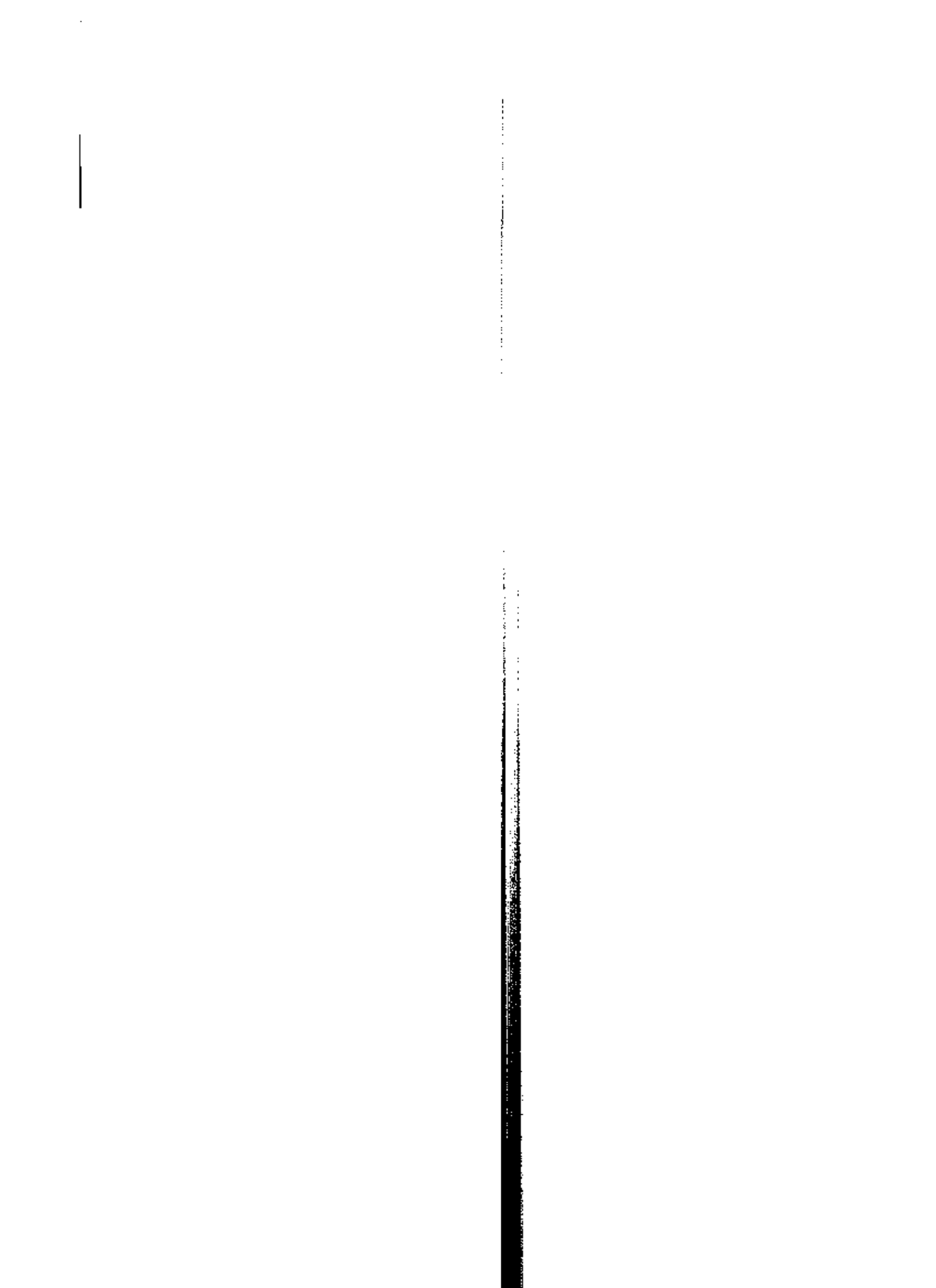
The mathematical method known as finite element analysis is widely used in the aerospace, automotive, and civil engineering industries. Engineers using this method analyze physical structures with a two- or three-dimensional grid that divides a structure into a number of well-defined elements. The effects of temperature- and pressure-related stress are then calculated for each element, as well as the effect of each element on its neighbors. Finite element analysis can be used to describe the overall impact of many loads and constraints acting on a complex structure. CRAY systems enable engineers to conduct rapid finite element analyses, resulting in improved engineering efficiency and more structurally sound and lightweight components. CRAY systems allow for detailed analyses of structures too large or complex to be analyzed on other computers.

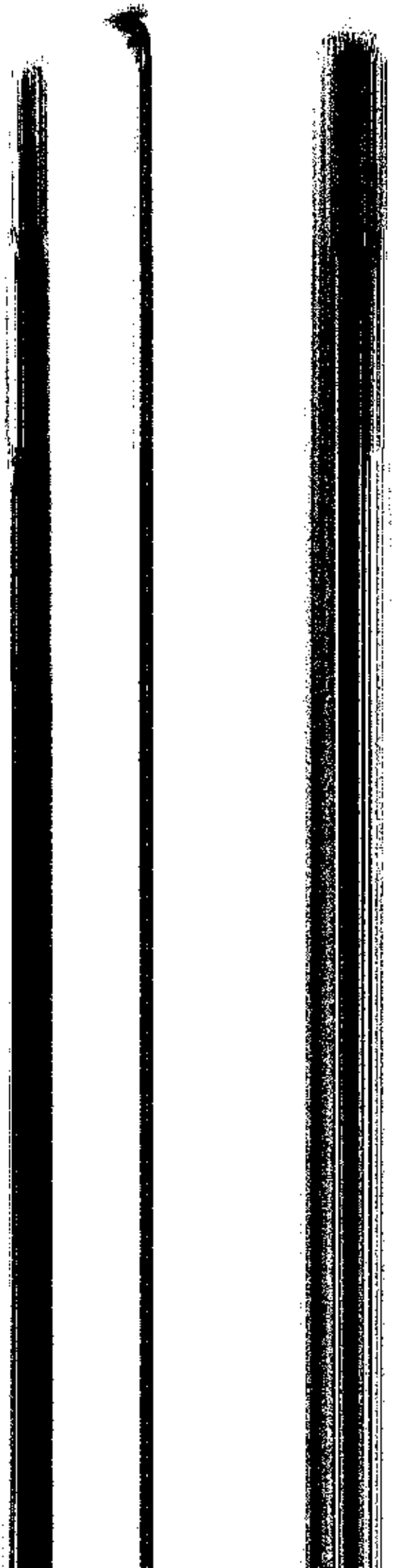
Meteorology

Meteorologic laboratories in the United States, Canada, and Europe use CRAY systems to model the Earth's atmosphere for weather forecasting and climatological research. To be useful, atmospheric models must accurately account for physical phenomena including evaporation and condensation, solar heating, and cloud movement. The models must also include information about numerous variables such as temperature, humidity, barometric pressure, and



licensing processes in several countries. CRAY systems used in nuclear energy research help to design reactors, implement automatic process control, train operators, analyze reactor conditions, and determine accident mitigation procedures. Safety analyses using computers are necessary since it is impractical to run accident experiments on full-scale nuclear reactors. Only CRAY systems provide the computing power needed to simulate the fluid flow, heat transfer, and neutronics phenomena that characterize today's nuclear power plants.





SECTION 13

Cray Product Familiarization

SCIENTIFIC COMPUTING OVERVIEW

Cray Research, Inc. Software Training

THIS PAGE IS INTENTIONALLY LEFT BLANK.

RATIONALE FOR SUPERCOMPUTING

based on notes from
"THE FUTURE OF SCIENTIFIC COMPUTING:
TRENDS IN SUPERCOMPUTERS"
by Jack Worlton

Note: Jack Worlton, of Worlton and Associates, Los Alamos, NM, is an expert in supercomputing technology and a consultant to Cray Research, Inc.

Cray supercomputers are very valuable in today's scientific computational environments, but what was the basic evolution to get to today's machines and are they fast enough for the future?

In the 1940's accounting machines achieved computational rates of about 1 or 2 floating point operations per second (FLOPS). Today's X-MPs achieve rates close to 100 Million FLOPS (MFLOPS) which amounts to an increase in power of 10^8. Another way of looking at it is **all the computing done in the '40's could be done in one second today.**

However, scientific computing requires faster and more efficient machines yet in order to model different and more complex phenomena. And just because computer memory has gotten larger and computing speed has increased drastically, other related technology has not kept up. Disks are still far too slow (rotational delay, e.g.). The SSD helps here. Archival storage is not satisfactory (tape, e.g.). Optical disk is a possibility, but there is no device yet and no standard has been established. Also, better connectivity (networking) environments need to evolve.

In the future, budgets for scientific computing will include PCs and supercomputers, not mainframes. PCs have improved immensely and supercomputers are so much more efficient than mainframes that the work to be done can be accomplished by the PCs and supercomputers alone. There already is an overlap of high-end mainframes and low-end supercomputers (e.g., IBM 3090 and Cray X-MP/14se).

EXAMPLES of SUPERCOMPUTER USES

Storing oil for future use is often done by creating and filling cavities in salt domes. The process involves leaching out an area of salt, but the shape of the caverns created cannot be predicted. This could pose a problem if several adjacent caverns were created - one could leach out horizontally to enter another cavern.

The Dept. of Energy (DOE) saved $230,000 per cavern using computer modeling and they did 100 of them for a total savings of $23,000,000.

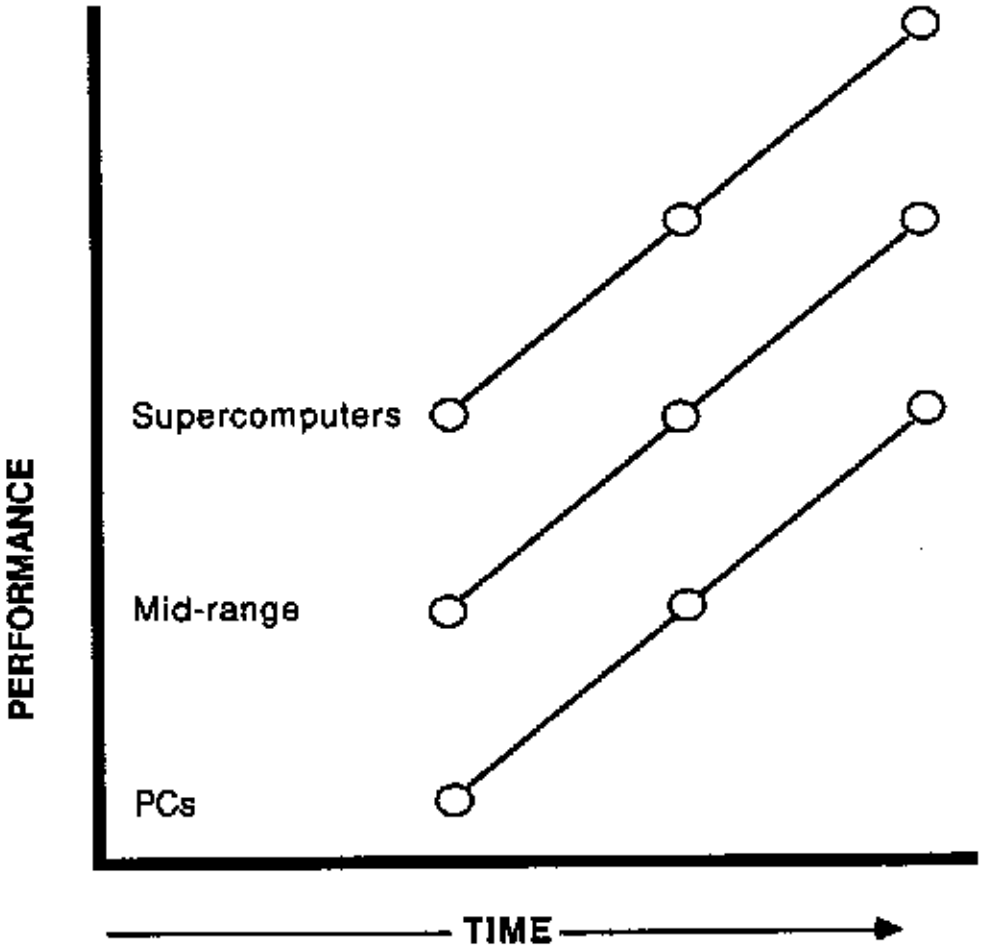
Sandia Labs needed to design a parachute which would open at low speeds, but be released from a high-speed jet aircraft. Without computer modeling 163 physical tests would have been required. With computer modeling only 105 tests were needed. Since each test cost $50,000, $3,000,000 was saved.

In an effort to minimize nuclear missile silo shock a computational fluid dynamics program was run on an X-MP at Cray Research, Inc. in Mendota Heights. The silo was subsequently redesigned and the new design saved $25,000,000.

Supercomputer testing is also valuable in cases where testing is just not feasible or safe. A better design was found for the release of bombs from a jet on a supercomputer. Certain conditions and designs could cause the bomb to flare up and strike the plane that released it. One design, for example, showed that a released object could hit the plane in 4/10ths of a second.

Even though supercomputers are expensive, their lifetime of usefulness is 10-15 years, and by giving results as in the above examples, they more than pay for themselves.

TRENDS IN PERFORMANCE



Rationale for Supercomputing

There are several reasons for using supercomputers, including the following:

Tractability - Supercomputers can solve problems which otherwise can't be solved in a practical time

Cost - It's cheaper than physical modeling or slower computers; they can save millions and even billions of dollars in research costs.

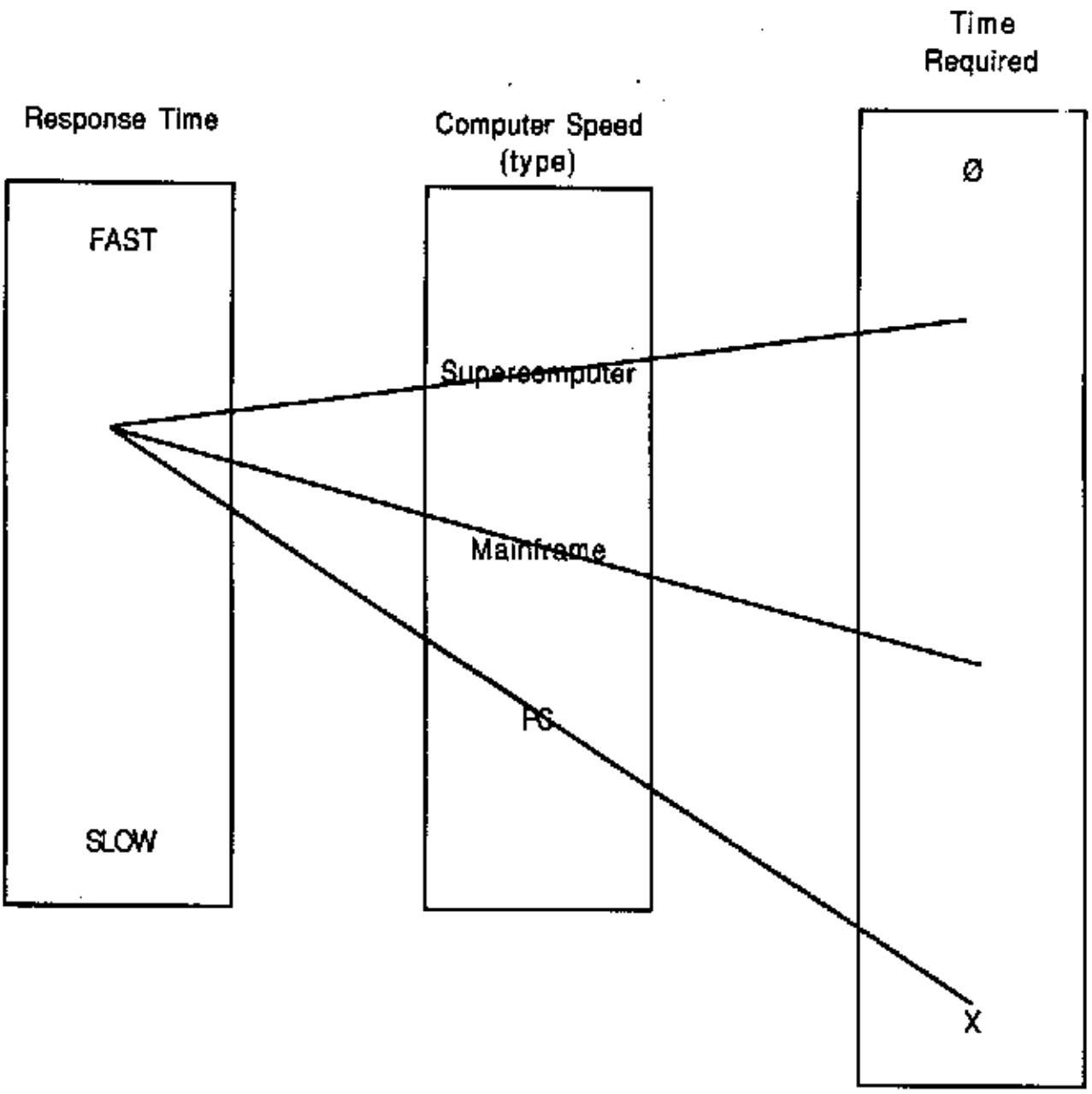
Time - Supercomputers are inherently faster

Quality - It's very easy to modify a program to test out different options

Productivity - The organization (personnel) becomes more productive accomplishing more in less time

WHY SUPERCOMPUTING?

Execution rate $\left(\frac{\text{operations}}{\text{time}}\right) = \frac{\text{complexity (no. of operations)}}{\text{response time}}$



SUPERCOMPUTER EVOLUTION

The distinguishing feature of a supercomputer is that it is at the high end of performance as measured by the number of instructions it can perform per second (floating point operations - FLOPs). The evolution in computer design which allows this began with **scalar** machines which basically interpreted one instruction and did one operation on one pair of operands.

Later improvements allowed analysis of a second instruction (or more) while the first was executing (**lookahead**). Then came **parallelism** which allowed many functional units to be working at the same time on different instructions (e.g., adders, multipliers, etc.).

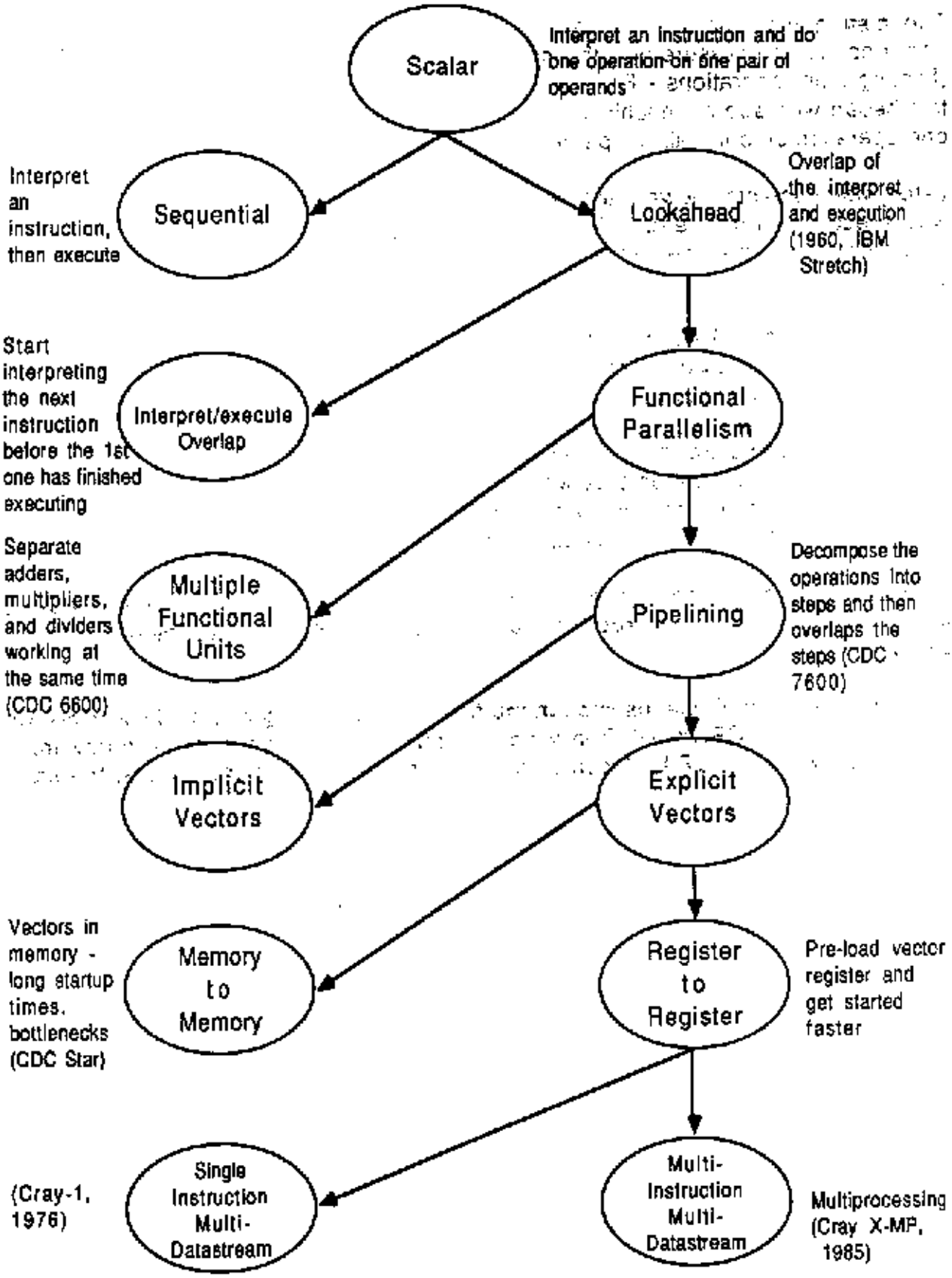
Later, **pipelining** broke down the functional units into segments so operations could be decomposed into individual steps (e.g., a second add operation could start before the first one was comp leted).

Vector operations allowed many related pieces of data (elements) to be loaded into an area while neighboring elements were being used as operands. This eliminated the need to load two operands, run them through a functional unit (e.g., add), then load two more operands and add them, as pure scalar operations require.

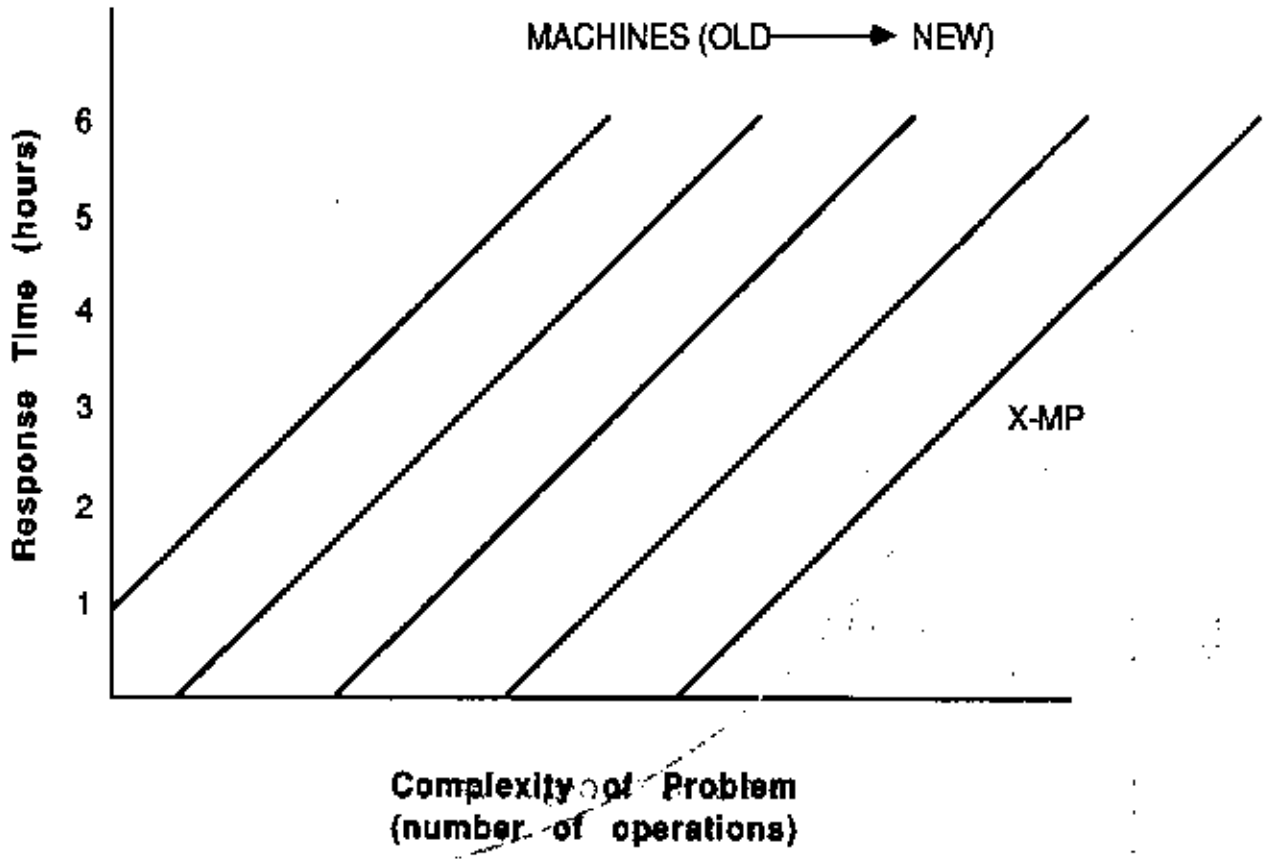
The combination of functional parallelism, pipelining, and vector registers made up what is known as **single instruction multiple datastream.** This was the design of the CRAY-1.

The technique for **multiple instruction multiple datastreaming** was made possible with the advent of the CRAY X-MP in which multiple CPUs (each with the hardware to accomplish what the CRAY-1 could) can work on separate instructions at the same time.

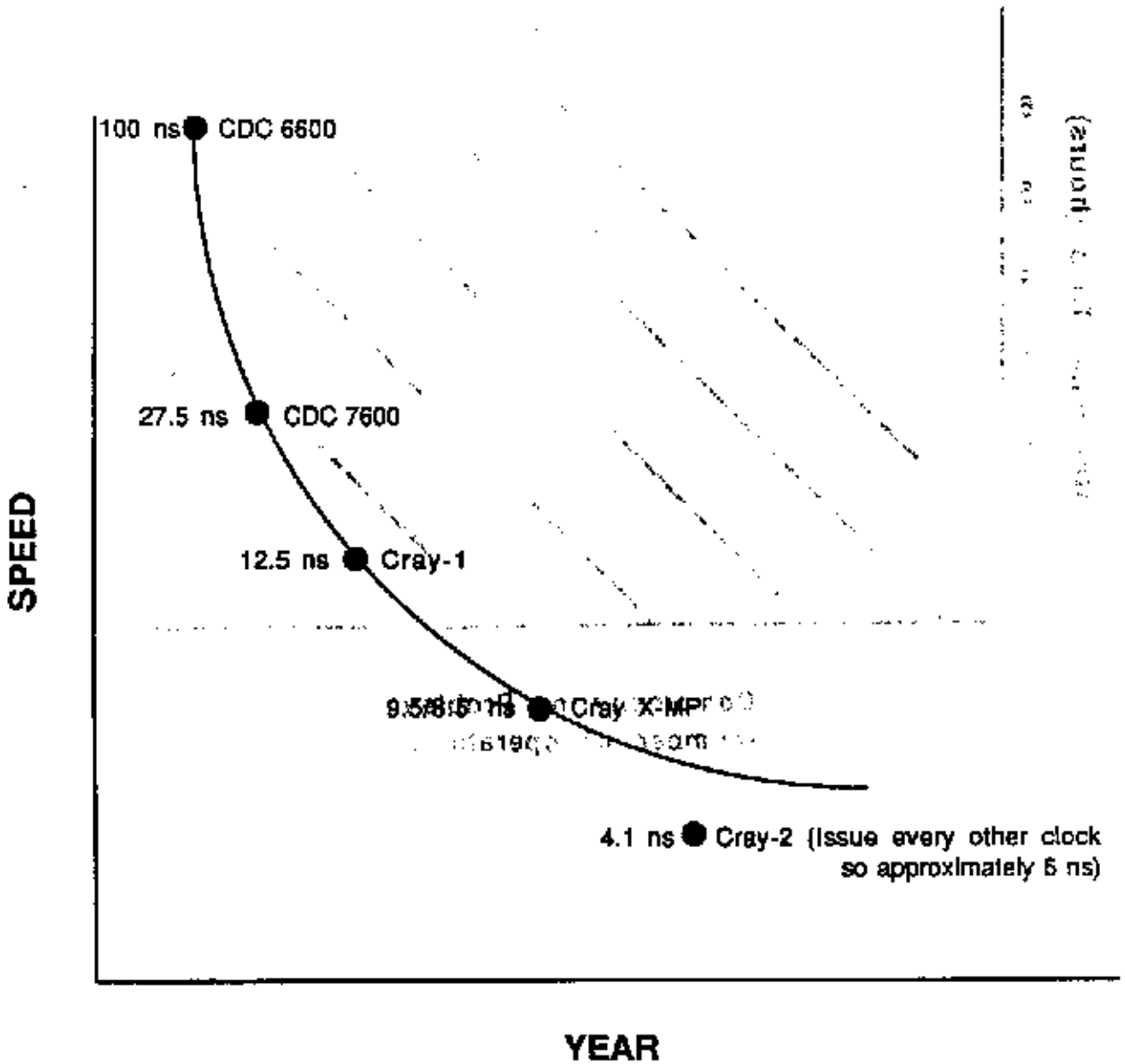
EVOLUTION OF SUPERCOMPUTERS



RESPONSE TIME vs. COMPLEXITY OF PROBLEM



SPEED-UP IN CYCLE TIME



SEMICONDUCTOR TECHNOLOGY

Over the years many different types of computer chips have been developed. Obviously, it is very important to supercomputer technology how fast the chips are and how much power they consume. Some of the current and new developments for supercomputer use include:

ECL - Emitter-coupled logic
This chip is curreptly used in many X-MPs. It is relatively fast, but requires a lot of power and generates lots of heat. It is based on silicon technology.

GaAs - Gallium Arsenide
This chip design is more efficient than ECL but does not have room for as many gates per chip, so more cycle time is needed to go from chip to chip. This was not a problem with silicon technology.

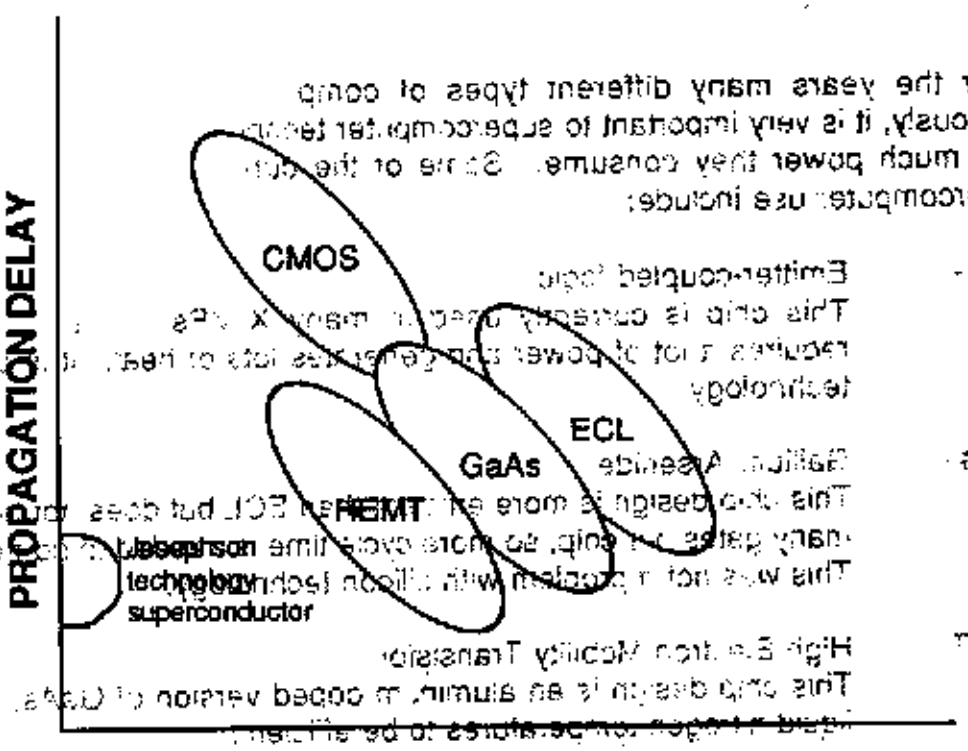
HEMT - High Electron Mobility Transistor
This chip design is an aluminum doped version of GaAs, but must run at liquid nitrogen temperatures to be efficient.

CMOS - Complimentary Metal Oxide Semiconductor
This chip design offers high component density. It appears to be slow but is very fast at supercooled temperatures. ETA uses this technology.

Josephson Technology Superconductor -
This chip design offers both speed and low power consumption. However, it has been shown to be difficult to mass produce and must operate at liquid helium temperatures.

Chip development efforts are moving toward more speed with less power consumption (toward the corner in the graph).

CHIP TECHNOLOGY



PREDICTED CHIP SPEEDS