CDC CYBER 200/MODEL 205

TECHNICAL DESCRIPTION

CONTROL DATA CORPORATION

CYBER 200 MARKETING SUPPORT DEPARTMENT

October 1980

CONTENTS

PREFACE

This document describes the basic hardware and software characteristics of the super-scale CDC CYBER 200/ Model 205 Computer System. Section 1 provides a general description of the system, including the basic architecture of the Model 205 and its hardware and software components. Section 2 covers the hardware of the Model 205 in more detail. A detailed description of the software is contained in Section 3.

Section 1

GENERAL INFORMATION

INTRODUCTION

Control Data Corporation is committed to development, manufacture and support of large-scale high-performance scientific computers. The CDC CYBER 200/Model 205 Computer System will satisfy the needs of users with high-performance computer requirements. This effort began with the design of the CDC STAR-100, continued with the identification several years ago of promising technology developments for high-performance computing and now provides enhanced memory and scalar processing performance and the development of faster vector processing and increased input/output capacity.

As early as 1973, Control Data began investigating Large Scale Integration (LSI) technology for use in large digital computers. Control Data also developed computer-aided design and simulation techniques which now enable LSI technology to be utilized in the largest digital system. The design of a 6400 Class LSI test vehicle was initiated in 1974; LSI components and design simulation tools were selected; and a working model was demonstrated in 1976. The success of these efforts led to further study and development in the areas of packaging and cooling technology. During this same timeframe, Control Data was also studying ways of expanding the performance of the CDC STAR-100 computer.

These activities led to a Corporate commitment in 1975 to design and implement semiconductor memory with an LSI scalar unit in the CDC STAR-100 and to replace the vector processor on the CDC STAR-100 with LSI circuitry. This new computer system is known as the CYBER 200 Series.

Control Data's commitment is to design, develop, manufacture and market a vector computer technology which substantially improves the CDC STAR-100 series in terms of reliability, performance and maintainability. The goal of the simulator and test vehicle construction was not to develop a new computer architecture but solely to prove the feasibility of new circuit technology that was not available when the CDC STAR-100 was developed. The proven architecture of the CDC STAR-100 system continues in the enhanced CDC CYBER 200 series computers.

The first member of the CYBER 200 Series was the CYBER 200/Model 203 which was announced early in 1979. CYBER 200/Model 203 deliveries started in the fourth quarter of 1979. The Model 203 featured a high performance scalar processing unit employing large-scale integration (LSI) circuit technology, a bipolar semiconductor memory of one-half to two million 64-bit words, and a vector processing unit utilizing medium-scale integrated circuitry.

The second model of the CYBER 200 Series Systems, the Model 205, features a new LSI vector unit integrated with the Model 203 LSI Scalar processing unit.

The Model 205 uses LSI circuitry throughout. High density chips (168 gates per chip), improved packaging, and subnanosecond switching time result in exceptional performance and high reliability. High levels of reliability and maintainability are enhanced by dense integrated circuitry reducing interconnects and a cooling system which maintains a low semiconductor junction temperature. Fewer than 30 different pluggable LSI circuit types are used in the central processor.

ARCHITECTURAL CONCEPTS

From a user's point of view, many operations in the Model 205 are performed in a serial fashion; other operations are performed in a parallel mode; but all operations are issued in strict sequence from a single instruction stream. The serial or scalar processing mode is common to almost all computer systems today. The parallel or vector processing mode allows the manipulation of many operands by a single instruction. Functional concurrency is permitted wherever possible while retaining logical integrity of the user's program.

In super-scale computer systems such as the Model 205 computer system, a distribution of the many tasks among specialized units is the key element for speed and for economic functioning. Within the Model 205 system, this is achieved by assigning various numerical, input/output, and nonnumerical operations to a variety of specialized sections and units such as input/output ports, the functional units of the scalar processor, and the stream, string and array sections of the vector processor.

The Model 205 is assigned primarily to the computational aspects, leaving input/output operations and many other support tasks to front-end and station computers. This "functional hardware concept" or distributive processing concept is the cornerstone of the Model 205 system architecture. Functional and distributed hardware means economy, controlled growth and expansion capability, and better overall availability and total performance. The Model 205 provides high performance in computing and in associated disk and high density (6250 cpi) tape input/output. The front-end computer can provide input/output for tapes, unit record equipment, and remote devices as well as data management and network control.

The Model 205 hardware supports floating point, integer, bit and byte and character data types. These data formats are demanded by many scientific applications and are supported by commonly used compilers and more advanced programming languages.

The extensive instruction repertoire allows computation and data processing both in the traditional serial or scalar processing mode, requiring at least one instruction to perform an operation on a single operand or operand pair, and in the vector processing mode, producing many arithmetic results on multiple operands or operand pairs. Vector processing is, in essence, a parallel processing mode. The speed of serial or scalar processing is dictated by the circuit speed and, hence, restricted by the speed of light. In vector mode, as many as 800 million 32-bit operations per second may be computed by the Model 205.

Vector processing is quite adaptable to many scientific and engineering computational processes where vectors are a common notion. The instruction set supports many simple and complex vector operations and even provides application primitives which must be expressed as program loops on other computers.

"More computation per issued instruction" and the parallelism of multiple functional units, which is transparent to the user, are the key architectural concepts in achieving the computational performance that the Model 205 system offers.

HARDWARE

SCALAR PROCESSOR

The scalar processor features multiple segmented functional units which are pipelined to accept new operands every clock cycle. A 64-word instruction stack permits many large loops of code, once loaded into the stack, to be executed without repeated memory references to reload the instructions. The 256 general-purpose registers are provided to greatly simplify programming as any register can be used for any operation. Many variables can be permanently resident in this register file. In the case of memory-resident data, the large number of registers and the design of the load/store unit combine to permit an essentially unlimited number of load (memory fetch) instructions to be issued with minimum time penalty, as necessary to complete the computations required within any program.

VECTOR PROCESSOR

The vector unit can operate in parallel with the scalar unit. Each operand in a vector operation can contain data in sets of up to 65,535 elements. Vectors are defined as contiguous sets of data elements. An array is a vector of floating point elements. A string is a vector of bit or byte elements. The control vector is a bit string used in the control of array operations. Control vectors make it possible to imbed decisions within array operations by inhibiting the store operation on selected elements of the array. Special vector operations such as dot product and square root provide the equivalent of complete subroutines on other computers. The benefit of these features is that more code can be vectorized without intervening scalar instructions, thereby simplifying programming and increasing performance.

MEMORY

The Model 205 provides both a large, real memory and essentially unlimited virtual memory. The hardware address space provides a virtual memory of 2 trillion words per user, eliminating the need for programmer concern about running out of space. (The 2 trillion words of virtual memory address space is a theoretical maximum. Actual virtual address space is limited by the number of CDC 819 disk drives connected to the system.) The Model 205 provides real memory sizes of 1 million, 2 million and 4 million 64-bit words. Memory can be addressed in full word, half word, byte, and bit units. To enhance reliability, circuitry is provided to correct single bit errors and to detect double bit failures for each 32-bit half word; thus, automatic correction for even two memory failures in a logical word will occur when the failures are in different half words.

INPUT/OUTPUT PORTS

Sixteen input/output ports can be connected through channel couplers to disk drives and front-end processors or stations. The purpose of the stations is to remove the burden of peripheral input/output from the central processor. The Model 205 central processor is devoted to computational tasks while the station processors handle device communications.

LINK HARDWARE

The hardware link between the Model 205 computer and the front-end station processors consists of the channel couplers. The channel coupler provides the compatible interface which allows direct connection of the data channels and controls the transfer of all data between the two mainframes.

SOFTWARE

Standard Model 205 software components include the operating system (CYBER 200-OS), CYBER 200 FORTRAN, META assembler, Loader, and link software.

OPERATING SYSTEM (CYBER 200-OS)

CYBER 200-OS is a multiprogramming operating system which provides a file system with security and back-up facilities, a full job and file recovery system, and a well-defined interface for computer-to-computer linkages. The operating system and FORTRAN combine to provide facilities which are adequate not only to support efficient production usage of the Model 205 but also to facilitate conversion and new code development; interactive and symbolic debugging capabilities are examples. Support codes include an assembler, a loader, source and object code maintenance utilities, and file utilities.

CYBER 200-OS is designed to support all Model 205 hardware features. The virtual memory implementation in the FORTRAN environment removes one of the most limiting factors imposed on scientific users. In this environment, all input/output operations can be performed implicitly by the system. However, where real-time limitations are of paramount importance, the user may employ explicit input/output calls which efficiently overlap computation and input/output for a given job. Another advantage of virtual memory is that large codes can be accommodated without the necessity for using an overlay facility.

To allow efficient multiprogramming and support, CYBER 200-OS was designed as a task-oriented system. System resources are shared among system and user tasks. System integrity is maintained through the use of virtual memory hardware protection.

CYBER 200-OS is file-oriented, with all jobs, tasks, and data existing as files. The file mechanism allows definition of selective read and write access to files. The file system defines files as private to a particular user, common to all users, or shared among a specific pool of users managed by a pool boss. The file technique is also an integral part of the recovery which the operating system provides since temporary and input files which were transmitted by front-end stations are not destroyed until after job execution and output files are retained until successfully transmitted to front-end stations.

CYBER 200 FORTRAN

CYBER 200 FORTRAN implements the standard FORTRAN (as defined by American National Standards X.9-1966, FORTRAN) with many extensions. These extensions already provide many of the capabilities that are in the FORTRAN 1977 standard. CYBER 200 FORTRAN also provides vector language extensions and direct access to all central processor instructions which makes efficient machine utilization possible without the necessity for assembly language programming. Efficient code development is aided by a symbolic cross-reference map and a symbolic debugging package.

CYBER 200 FORTRAN takes advantage of the hardware capabilities by using the registers for intermediate operands and results, vector descriptors, and particularly for FORTRAN scalar variables. At the time that a subprogram is invoked, local variables are block-transferred to the register file where they may be rapidly accessed and updated. In this way, the potential for high performance, which is inherent in the large general-purpose register file, is realized by the CYBER 200 FORTRAN implementation which automatically assigns most scalar variables to registers during their active life.

META ASSEMBLER

META is the assembly language for the Model 205 central processor. The assembler generates relocatable binary output which is linked and loaded by the Loader under operating system control. META provides:

- Conditional assembly capability for selective assembly.
- Set capability to define, reference, and extend the list of expressions.
- Procedure and function capability.
- Attribute assignment for symbols and elements.

LOADER PROGRAM

The Loader program provides the user with a means of collecting and linking relocatable programs and subprograms to produce an executable program. The final product is a file ready for execution under control of the operating system.

LINK SOFTWARE

Control Data has software packages available for controlling the hardware link from the Model 205 computer to front-end processors and high speed disk and tape stations. These packages operate in conjunction with the Model 205 operating system (CYBER 200-OS) and the front-end processor's operating system. CDC CYBER 170 series mainframe running under the Network Operating System (NOS) or the NOS Batch Environment (NOS/BE) system are examples of systems used to front-end the Model 205. This approach offers the availability of each system's software product set to the programs residing in the other system.

Section 2

HARDWARE DESCRIPTION

CDC CYBER 200 MODEL 205 COMPUTER OVERVIEW

The Model 205 computer is a super-scale, high-speed, logical and arithmetic computing system. It utilizes LSI circuits in both the scalar and vector processors that improve performance to complement the many advanced features that were implemented in the STAR-100 and CYBER 203, such as stream processing, virtual addressing, and hardware macroinstructions. The Model 205 contains separate scalar and vector processors specifically designed for sequential and parallel operations on single bits, 8-bit bytes, and 32-bit or 64-bit floating-point operands and vector elements. The central memory of the Model 205 is a high-performance semiconductor memory with single-error correction, double-error detection (SECDED) on each 32-bit half word, providing extremely high storage integrity. Virtual addressing uses a high-sped mapping technique to convert a logical to an absolute storage address to allow programs to appear logically contiguous while being physically discontiguous in the storage system.

The basic Model 205 computer consists of the central processor unit (CPU), 1 million 64-bit words of central memory with SECDED, 6 input/output ports, and a maintenance control unit (MCU). The CPU contains the scalar processor and a vector processor with one vector pipeline. Central memory is field-expandable from 1 million 64-bit words to 2 or 4 million words of semiconductor memory. The vector pipelines can be expanded to 2 or 4 and the input/output ports are expandable to 16.

The Model 205 central processor contains all instruction and streaming control, scalar and vector arithmetic processors, and control for communication with central memory by the CPU and the input/output channels.

Figure 2-1 shows the basic functional areas of the Model 205 CPU:

- Scalar Processor
- Vector Processor
- Memory Interface
- Maintenance Control Unit

The physical layout of the CPU and the central memory is shown in Figure 2-2.

The LSI scalar processor contains a scalar arithmetic unit with independent high-speed scalar arithmetic functional units. The scalar processor also contains a semiconductor register file of 256 64-bit words used for instruction and operand addressing, indexing and storing constants and field length counts, in addition to holding operands and results for scalar instructions. The scalar processor performs instruction control and virtual address comparison and translation. A feature is provided to select, via an operating system software installation parameter, a small page size of 512, 2048, or 8192 words. A large page size of 65,536 words is also provided.

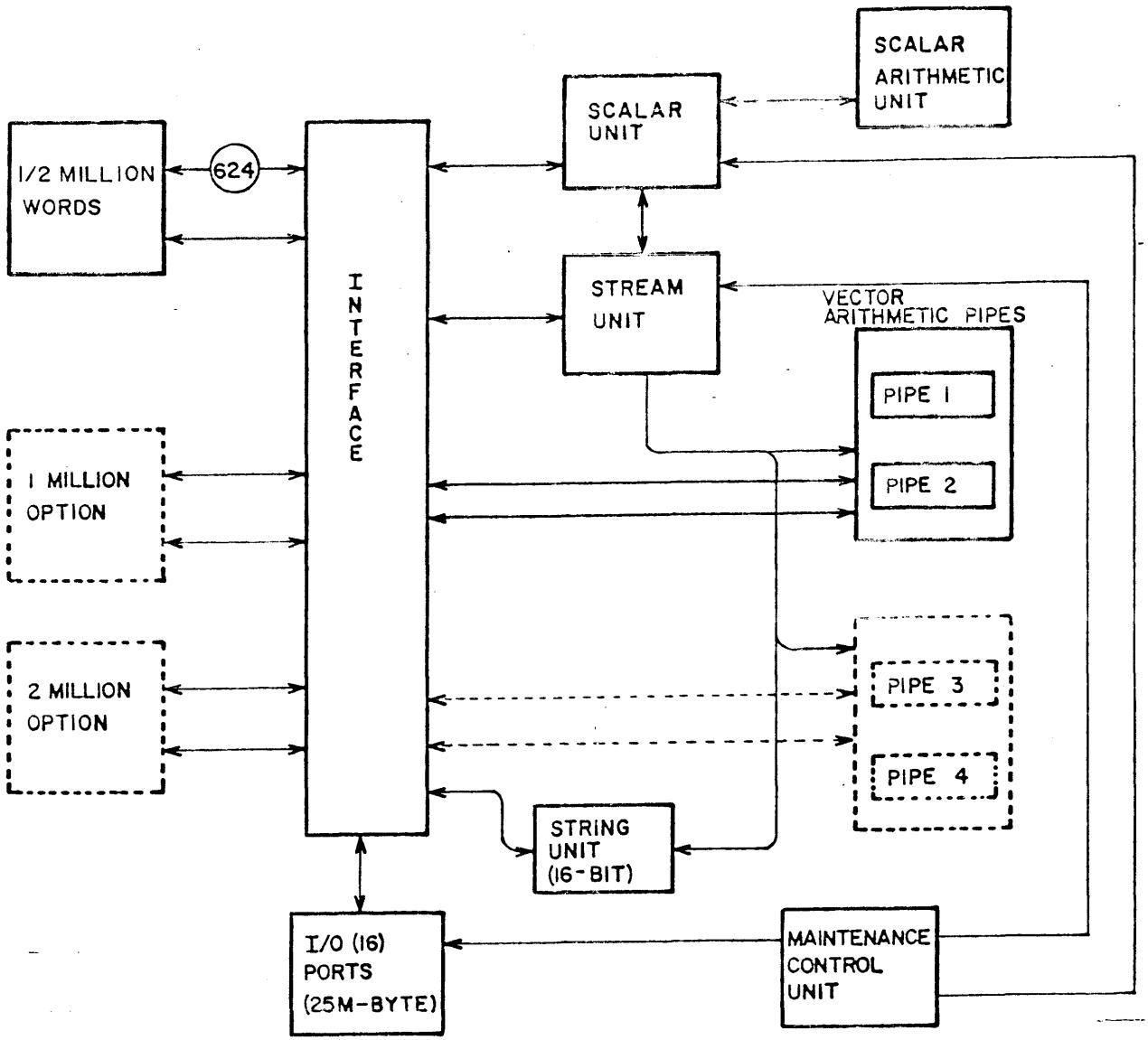
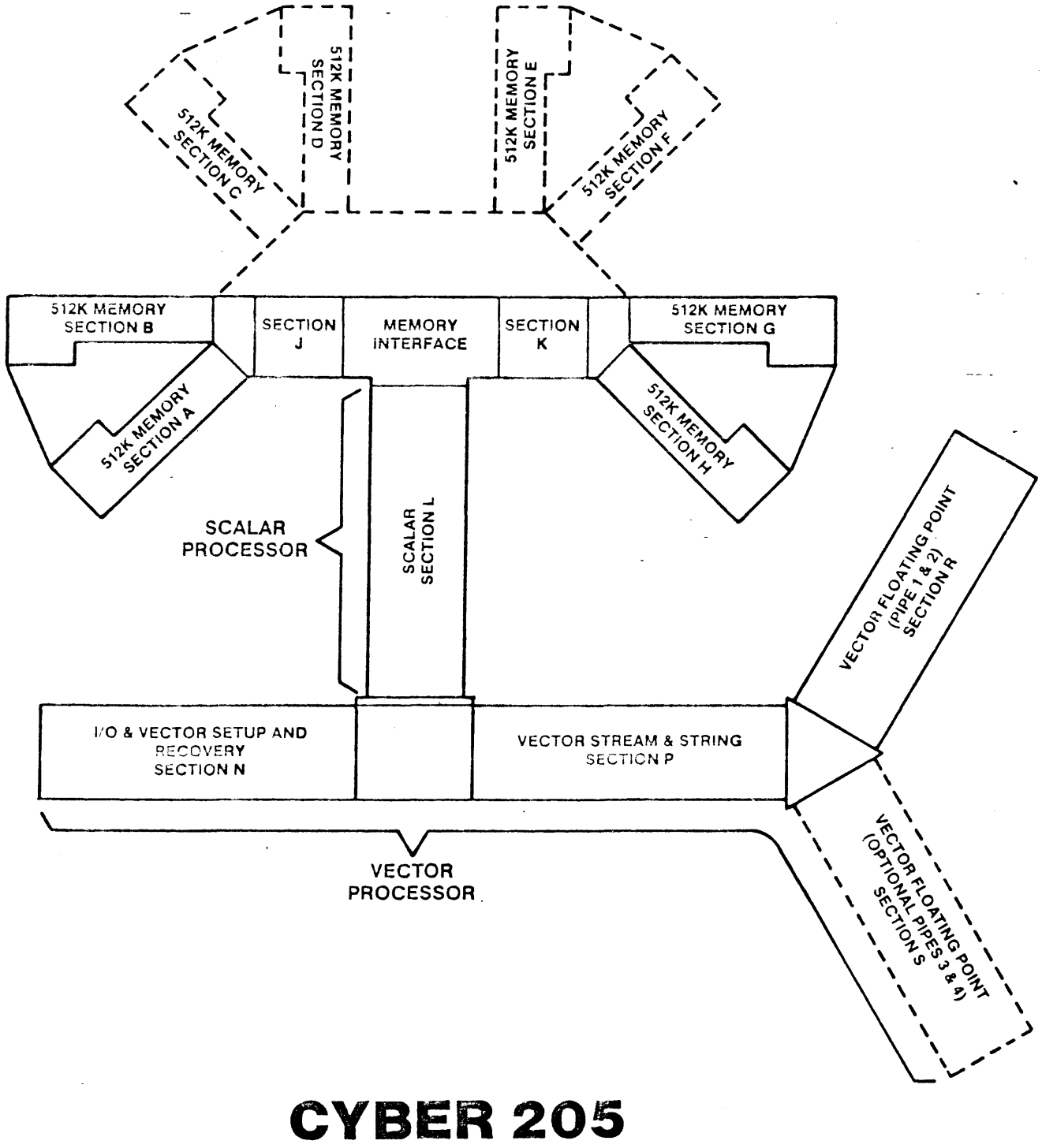


Figure 2-1. Model 205 Computer System Configuration



CYBER 205

Figure 2-2. Model 205 Computer System Floor Plan

The vector processor contains one, two or four parallel, segmented pipelines to facilitate high-speed vector processing. The vector processor control is contained in the stream unit. The string and all logical operations are performed in the string unit.

The memory interface provides the read and write ports of central memory for the scalar and vector processors. Each port contains a one-SWORD (512-bit Super WORD) buffer to facilitate high transfer rates.

The CPU processes input and output by issuing relatively simple high-level messages to high-speed peripheral stations or a front-end processor connected to the input/output ports.

SYSTEM CHARACTERISTICS

The general characteristics of the Model 205 hardware are summarized below. The hardware characteristics are described in detail in the following sections.

CPU Characteristics

- Minor cycle of 20 nanoseconds for both scalar and vector operations.
- Two's complement arithmetic.
- One, two or four parallel vector pipelines.
- Hardware macroinstructions.
- Sequential stream processing.
- Bit, byte, half-word, or 64-bit word floating-point operations.
- Independent scalar and vector instruction execution for no-conflict operations.
- Semiconductor high-speed register file - 256 64-bit registers (two reads and one write per clock period).
- Sixty-four 64-bit word instruction stack for the optimization of programmed scalar loop iteration.

Virtual Addressing Mechanism

- 48-bit virtual address. Actual virtual address space is limited by the number of CDC 819 disk drives connected to the system.
- Program protection via lock and key.
- 16 registers for simultaneous virtual to physical mapping.
- Selectable page sizes - small page sizes of 512, 2048, and 8192 words and large page size of 65,536 words.

Extensive Instruction Repertoire

- 32-bit and 64-bit floating-point arithmetic.
- Vector and sparse vector.
- Vector macros.
- Dot product.
- Square root instructions.

Reliable Central Memory Structure

- Semiconductor memory with 80-nanosecond access time.
- SECDED for each 32 bits for high reliability.
- Memory sizes of 1, 2, and 4 million 64-bit words.
- High memory bandwidth of 512 bits per 20-nanosecond minor cycle for a 1 million word system and 1024 bits per 20-nanosecond minor cycle for a 2 million word system, and 2048 bits per 20-nanosecond minor cycle for a 4 million word system.

Flexible High-Speed Input/Output

- 6-16 input/output ports.
- Each port is capable of 200 million bits per second maximum transfer rate.
- Connection to a CDC 6000 or CYBER 170 front-end computer (and some non-CDC computers) in a computational facility configuration.
- One channel is used for the maintenance control unit (MCU).

SYSTEM ARCHITECTURE

Central Processor

The Model 205 CPU contains a scalar processing unit and a vector processing unit.

Scalar Processor

In addition to providing for the execution of scalar operations, the scalar processor performs the primary system control functions of the Model 205.

The scalar unit contains a sixty-four word discontiguous instruction stack segmented into eight super-words (SWORDs). The instruction stack is capable of holding up to 128 32-bit Model 205 instructions, 64 64-bit

instructions or a combination of both, and provides a sixteen word instruction read ahead. The instruction issue pipe decodes all instructions, initiates scalar operations with the appropriate functional unit, and directs decoded vector/string instructions to the vector processor for execution. The instruction issue pipe is capable of issuing instructions at the rate of one instruction every 20 nanoseconds. Thus, with independent vector and scalar instruction controls operating on a single instruction stream, the scalar processor can execute scalar instructions in parallel with most vector instructions provided there are no memory references generated by the scalar instructions. A block diagram of the functional components of the scalar processor is shown in Figure 2-3.

The load/store unit provides special handling of the load and store instructions. The unit acts as a pipeline and is capable of initiating one load every minor cycle or one store every two minor cycles, provided a memory busy, access interrupt, or register file write-bus busy does not occur. A circular buffer containing six registers provides buffering for up to six load requests, three store requests, or a mixture of loads and stores.

The load/store unit is capable of loading a randomly accessed word of data from central memory in the register file in 300 nanoseconds after reading the base address and item count of the data. This time assumes a memory busy, access interrupt, or register file write-bus busy does not occur. A memory busy would add 80 nanoseconds to the load time.

The scalar arithmetic unit contains completely independent functional units to attain high scalar performance. Table 2-1 contains the times in nanoseconds to produce a 32-bit or 64-bit result in each functional unit. These times correspond to the short-stop times. Short-stop is the process by which a result from any arithmetic unit may be returned directly to either input of any arithmetic unit. This occurs in parallel with the storing of the result in the register file. Short-stop eliminates the time necessary to store the result in the register file and then retrieve it for use in the next arithmetic operation.

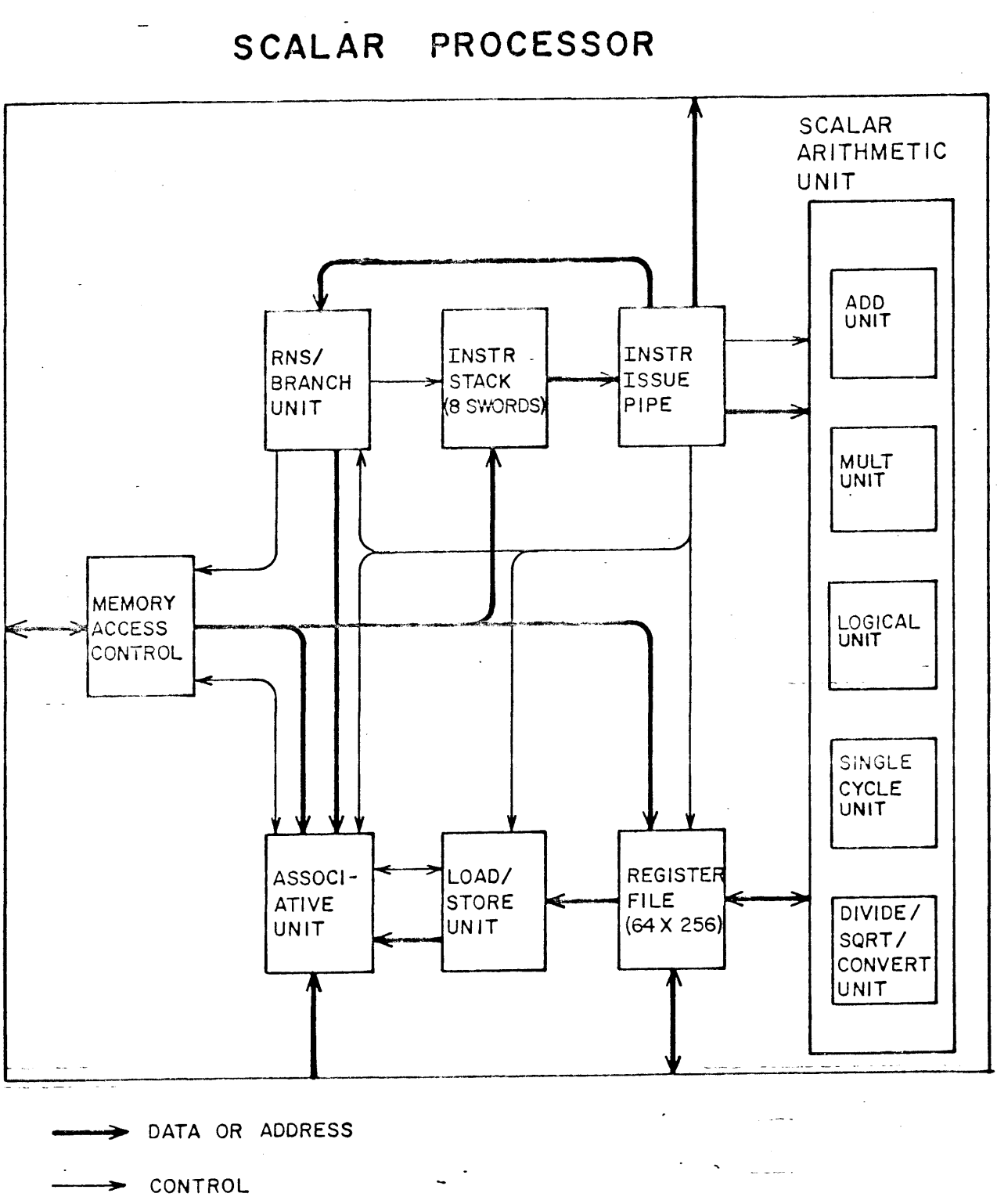


Figure 2-3. LSI Scalar Processor Block Diagram

TABLE 2-1. SCALAR PERFORMANCE

SCALAR UNIT	MINOR CYCLES ISSUE	SHORT-STOP	REGISTER FILE WRITE
INCREMENT/INTEGER	1	1	4
SHIFT/LOGICAL/PACK	1	3	6
FLOATING POINT ADD/SUBTRACT	1	5	8
FLOATING POINT MULTIPLY	1	5	8
FLOATING POINT DIVIDE	1	54	57
FLOATING POINT SQUARE ROOT	1	53	56
LOAD	1	—	15
STORE	2	—	—
BRANCH FALL THROUGH	8	—	—
BRANCH INSTRUCTION-STACK	9	—	—
BRANCH OUT OF INST-STACK	24	—	—
BRANCH TO SUBROUTINE (CALL)			
INTO INSTRUCTION STACK	8	—	—
OUT OF INSTRUCTION STACK	23	—	—

1 MINOR CYCLE = 20 NANOSECONDS

SHORT-STOP = "EXPRESS" TIMING FOR FIRST FUNCTIONAL UNIT USE OF RESULT

REGISTER FILE WRITE = TIMING FOR LOAD, STORE AND SUBSEQUENT FUNCTIONAL UNIT USE

The functional units are segmented and capable of accepting new operands every 20 nanoseconds except for the Divide/SORT/Convert Unit which must complete each operation before a new one can begin. All units are capable of being short-stopped.

The scalar processor contains a semiconductor register file which provides 256 64-bit registers for use in instruction and operand addressing, indexing, field lengths, and as source and destination registers for scalar instruction operands and results. The register file is capable of two reads and one write every 20 nanoseconds.

The Model 205 virtual memory feature provides the facilities for exploiting advanced techniques of memory management and user program protection. Some of these features are:

- Key and lock provide memory protection and user separation.
- Hardware mapping from virtual to physical addresses.
- An ordered page table to minimize operating system overhead.
- Program overlays at execution time formed by the hardware system transparent to the user's program.
- Sharing of user programs or data with other users.
- Small page sizes: 512, 2048, or 8192 64-bit words selectable by an operating system software installation parameter. The default is a small page size of 512 words.
- Large page size: 65,536 64-bit words.

The associative unit in the scalar processor contains the page table virtual addressing mechanism which is composed of 16 associative registers and a space table (located in a restricted area of central memory) with sufficient entries for up to four million words of central memory.

The page table is an ordered list of associative words necessary to define the pages in absolute memory. The 16 associative registers allow for immediate transformation of the most recently used 16 page table entries into real memory addresses. The ordering is done so that the most frequently used associative words tend to stay at the top of the table. The space table is an extension of the page table containing associative words necessary to define pages in absolute memory that have not been in recent use. The associative unit is capable of comparing all the associative registers in one clock cycle and the space table entries at the rate of two entries per clock cycle.

The paging mechanism of the Model 205 plus the operating system software permit the most active portions (pages) of a user program to reside in central memory. The virtual addressing facility, through the page table, makes these areas of physical memory appear to be contiguous in a manner invisible to the user. The paging mechanism ensures that a large number of users can have simultaneous access to the Model 205 computer, with minimum page swapping overhead.

Vector Processor

The vector processor unit consists of an operand streaming unit to control, buffer, and manipulate data to and from memory, a string unit for processing bit and byte operands, and general purpose segmented vector functional units called vector pipelines. The Model 205 has one vector pipeline in its basic offering and an option to increase the number of vector pipelines to two or four.

The vector processor functional unit is an integral part of the CYBER 200 Model 205 central processor, and designed to process vector hardware instructions issued by the scalar unit. The vector processor is capable of manipulating vectors with 1-bit elements, 8-bit elements, 32-bit floating point elements and 64-bit floating point elements.

The vector processor achieves extremely high performance based on its segmented operations synchronized with the basic 20 nanosecond clock period of the system and its method of streaming vector source and result operands directly from and to the broad bandwidth memory.

Additional refinements in the performance and capability of many of the CYBER 200 vector instructions, particularly those associated with the referencing of non-contiguous data, has provided the CYBER 205 with performance improvements over the earlier CYBER 203 significantly beyond the factor of two through the use of the LSI technology. These improvements include a 20 nanosecond versus 40 nanosecond clock cycle and up to four vector pipes.

Mega FLOPS (Millions of Floating Point Operations Per Second) performance rates attainable with the CYBER 205 vector processor are depicted in Table 2-2.

TABLE 2-2. PEAK VECTOR INSTRUCTION PERFORMANCE

	FLOATING PT. OPERAND SIZE	PEAK PERFORMANCE (MEGA FLOPS)		
NUMBER OF VECTOR PIPELINES	32-BITS AND 64-BITS	1*	2	4
VECTOR ADD/SUBTRACT	32-BITS	100	200	400
	64-BITS	50	100	200
VECTOR MULTIPLY	32-BITS	100	200	400
	64-BITS	50	100	200
VECTOR LINKED MULTIPLY AND ADD OR SUBTRACT	32-BITS	200	400	800
	64-BITS	100	200	400
VECTOR DIVIDE/SQUARE ROOT	32-BITS	15.3	30.6	61.2
	64-BITS	8	16	32
VECTOR DIVIDE/SQUARE ROOT (HIGH SPEED OPTION)	32-BITS	—	61.2	122.4
	64-BITS	—	32	64

*FOR MODEL 205-411 ONLY

Memory Interface Unit

The memory interface unit provides five ports for access to central memory. The scalar processor, vector processor and input/output ports are connected to central memory through this unit. The control of all data transmissions is provided by the memory access control in the scalar processor. Single Error Correction and Double Error Detection (SECDED) of each 32-bits of the data on the memory ports is performed in the scalar processor, and SECDED checking of vector operands is performed in the vector processor.

Data can be transferred to and from the memory ports in 32-bit half words, 64-bit words, or 512-bit SWORDS.

Each of the memory ports is connected to memory through a one SWORD buffer located in the memory interface. Where a buffer is shared by multiple ports, the memory access control provides proper port selection to the memory interface selection networks. Data is transmitted to and from the buffers in quarter swords at the rate of one-quarter sword per minor cycle (20 nanoseconds).

Central Memory

Central memory is a single-level, random-access memory using bipolar integrated circuits. The memory words are 78 bits which provide a 64-bit data word and 14 bits of SECDED (7 bits for each 32-bit half-word). The semiconductor memory access time is 80 nanoseconds. This memory is directly addressable in monitor mode and via hardware virtual relocation in job mode.

The basic central memory size is one million words with expansions to two or four million words available as field upgrade options.

Each one million words of central memory contains 16 memory modules each having 128 K 39-bit half-words (32 data bits plus 7 SECDED bits). Each module is arranged in eight phased banks. In streaming mode, a reference is made simultaneously to the same address in each of the 16 memory modules to obtain a super word (sword) of 512 data bits. Memory busy conflict rules take into account the 16 physically independent modules and the eight-bank phasing within each module to treat the bank address in each of the 16 modules as a separate entity. Thus, each million words of central memory contains 128 phased half-word banks when utilized in 32-bit mode and 64 banks as utilized in 64-bit mode.

The eight-bank phasing plus the physical distribution of the memory modules allows memory references to be made at a maximum rate of one every 20 nanosecond clock cycle. Thus, central memory has a high data transfer bandwidth of 512 bits per minot cycle in one million word configuration as required to support the operand request rate and the result request rate for two vector pipelines.

Input/Output Ports

The CYBER 205 has optionally 6, 8, 10, 12, or 16 Input/Output ports, each of which has a transfer rate of up to 200 megabits. Since the high speed transfer of data is so frequently a critical element in the effective execution of applications requiring a super-scale system, the CYBER 205 is designed to support the full bandwidth of data movement from all of these channels to central memory while vector and scalar operations are proceeding at their maximum rate. Also, the high individual bandwidth of each

channel provides excess capacity for future incorporation of new higher performance secondary storage devices as their technologies advance.

MAINTENANCE CONTROL UNIT

An important element contributing to maximization of reliability and availability of the Model 205 computer is the Maintenance Control Unit (MCU). The MCU, through hardware interfaces and sense lines together with sophisticated maintenance diagnostic programs and a complete set of interface tools for the maintenance engineer, provides a full range of maintenance and monitoring activities. The MCU is a dedicated Model 205 station providing a focal point for maintenance activities of the CPU. The MCU provides monitoring/logging/recovery of CPU faults, control of the CPU diagnostic system, management of CPU microcode memory, and system deadstart.

DISTRIBUTED PERIPHERAL NETWORK

Control Data utilizes a distributed peripheral network, specifically designed to solve the problems of interprocessor and peripheral connections and to provide an efficient approach to interfacing a variety of mainframes and peripherals in a local network. Control Data's solution to the local networking problem encompasses not only the hardware interconnection media, but also comprehensive software and diagnostic capabilities.

The system is comprised of several different hardware components. The transmission medium in the system includes one or more coaxial cable data trunks. These trunks provide reliable communication between attached devices at rates up to 50 million bits per second at 1000 feet. Communications over these trunks at longer distances are possible by reducing the number of attached devices. Each trunk has the capability of supporting multiple drops, thus providing for interconnection of many devices in a common network. Access Devices are used to interface various devices to the coaxial trunk network via a 50-megabit per second Data Set.

The architecture of the peripheral network is a generalized I/O system for multiple levels of independent communication control between multiple computer systems and/or peripheral devices attached to the computer systems. Each level has a well-defined functional task and well-defined interfaces to the adjacent levels. Three functional communication levels have been defined: data set communications, trunk interface communications, and attached device/processor communications.

The lower level of communication occurs between the data sets. The primary responsibility of the data sets is to transport data bits between trunk interface units via the transmission medium. The interface between the data set and trunk interface unit is defined so as to isolate the data set characteristics from the trunk interface unit logic. This defined interface will allow improved coaxial or other kinds of transmission systems (for example, light fibers, cable television, or microwave) to be used when appropriate.

The next higher level of communication occurs between trunk interface units. The primary responsibility of the trunk interface unit is to transport data and control messages and responses between the buffer areas of the Access Devices using the data set communications link. Since the transmission medium may

be shared among several data sets, a predefined trunk protocol must be followed. The trunk interface unit is responsible for generating and checking the channel protocol as well as maintaining the channel integrity.

The highest level of communications occurs at the attached device level. The Access Device processor is responsible for establishing, maintaining, and closing communications links between attached devices.

Components of the Distributed Peripheral Network hardware and software are utilized by the CYBER 205 system to provide appropriate interfaces between peripheral devices (such as the 819 disk subsystem or 6250 CPI tape) and the 200 megabit ports of the CYBER 205. It also provides the interface to other computer systems providing the front-end function to the Model 205.

Section 3

SOFTWARE DESCRIPTION

CDC CYBER 200 MODEL 205 OPERATING SYSTEM CHARACTERISTICS

The Model 205 Operating System, called CYBER 200-OS, was designed to provide the user with convenient access to the CYBER 200 computational facility either from a remote batch or interactive terminal or peripheral devices via the front-end computer. The objective of the total system is to make available to the user the computational capability and the many advanced Model 205 hardware features combined with the large product set available on the CDC CYBER 170 series product line. CYBER 200-OS accommodates the large multiprogramming base during one time of the day as well as the very large production jobs run at another time during the day. Such an environment is encountered in many industry areas such as atomic/nuclear, reservoir simulation and seismic data processing, numerical weather prediction and meteorological research, and structural analysis.

The operating system is the result of many years of design and development effort. Since mid 1974, the software system has been employed in a user's environment in which all features and aspects of the system have been explored and extensively tested. Enhancements have been provided since the initially released version for the CDC STAR-100, and additional enhancements are planned to support new hardware capabilities and features offered in the CYBER 200 product line.

The set of software available with the Model 205 computer system, ranging from software and file maintenance aids to basic mathematical functions, may be categorized in the following groups:

- CYBER 200 operating system
- Language processors; FORTRAN and META Assembler
- Software maintenance aids
- External CYBER 200-OS characteristics

CYBER 200-OS permits concurrent and parallel operations of many computational and input/output activities. The user interfaces with the system via the front-end computer. The CDC CYBER 170 series front-end provides access via interactive or remote batch terminals as well as through local unit record equipment. Information is passed between the front-end computer and the Model 205 computational facility using the distributed network described in Section 2. Any required data conversion of binary and coded information is handled by the access devices within the network prior to transmission, between computer systems.

Since a distributive concept was the major guideline in the hardware design of the CYBER 200 computer, a distributed system and processing approach is also the most logical choice in the software design to achieve economic functioning within the total CYBER 200 data processing complex. Thus, a major attribute of the software system is its high modularity and task-oriented structure, based on an efficient file concept.

Major characteristics and features of CYBER 200-OS include:

- File oriented system with security and backup facilities based on high performance random access mass storage.
- Support of virtual memory addressing.
- Input/output may be performed implicitly or explicitly.
- Implementation of a multi-tasking concept.
- Extensive accounting and resource utilization information.
- Job and file recovery.
- User and system checkpoint/restart.

FILE-SUBSYSTEM

A portion of CYBER 200-OS with which the user is very concerned is the file system. There are the following two major aspects in looking at the file system.

- A differentiation of the files based on the file ownership category.
- A differentiation based on the method in which the input/output operation is performed.

Based on the first aspect, CYBER 200-OS recognizes three file ownership categories. The user has the following access capabilities under each:

- Public Files

  The public files contain assemblers, compilers, and other general purpose routines and utilities. These files are managed by the system's privileged users who have Read/Write/Execute access to these files. These files are available to all users with the access rights granted by the system privileged user, generally Read and Execute.

- Shared Files

  Shared files or pool files are accessible to a subset of users as specified by the owner, user or administrator of the pool. The owner specifies the type of access (any combination of Read/Write/Execute). The members of the pool can access a pool file with any access that is granted by the administrator.

- Private Files

  Private files can only be accessed by the originating user. They may be local files existing only for the duration of a job or terminal session, or permanent files which are retained between jobs or sessions.

The second aspect in looking at the file system is based on the file input/output method. To perform file input/output operations the user can choose between two methods:

- Implicit input/output supported by the virtual memory system leaving the data manipulation and "overlay" handling task up to the operating system and supporting hardware.

- Explicit input/output to provide the user with maximum control over essential system resources during the execution of certain time dependent tasks.

Since CYBER 200-OS supports the virtual memory hardware concept allowing the user to address more than 2 trillion ($2X10^{12}$) 64-bit words directly, the user does not have to consider overlay techniques. The operating system manages allocation of storage between main memory and mass storage, moves information from mass storage to main memory as needed, and, by setting values in the page table, permits the hardware to translate virtual memory addresses to physical addresses in main memory.

CYBER 200-OS considers every program to be executable only in virtual memory which means that every page in virtual memory has a corresponding space on the disk storage system. Executable code exists first as a file on mass storage. This file, called a program file, is built by the loader and contains a map at the beginning of the file called the minus page. This map corresponds to the virtual page addresses and relates them to corresponding logical disk addresses.

The program file exists as read only, and the integrity of the program file is never violated. At the beginning of execution the system automatically creates another file, called the drop-file. The purpose of the drop-file is for program swap-out by the system and allows the user to terminate at any time and reconvene at any time. Also, the drop-file can be saved and used for recovery purposes. The drop-file is automatically updated by the system to contain memory pages containing any program data modified during program execution. If a program is modified dynamically during execution, the modified portions are swapped in and out from the drop-file; the original program file is never modified. If a user wishes to modify the program file, then a new program file must be created to contain the modified program. The user may then release the original file.

The code and the data of the program need not be in contiguous physical memory locations. The original user code is kept on the program file which is swapped into virtual memory when required but never swapped out to that file since it allows "read-only" access. The data associated with the program file can be swapped in and out of memory as necessary to execute the program. The swapping of program code as previously mentioned occurs with the drop file so that the integrity of the program-file is maintained. By allowing "read-only" access to the program-file, it can be executed over and over again by the user. The drop-file provides a snapshot of the execution stage in the program and can be used as an aid in debugging the program. CYBER 200-OS provides a checkpoint restart capability consisting of saving the drop-file and restarting the saved file.

During program execution the program can either open existing file space or create new file space and assign the type of access it desires to that file. It can assign a read-only type access to the file in which case the programmer would be notified if any attempt were to be made to write into that file. It can assign a read/write access to the file in which case the data is swapped into memory and back to the file. It can also assign a type of access termed write-temporary access in which case the data file is treated like program-files; that is, pages that are modified are never swapped back to the original file, but rather to the drop-file. The user never needs to issue any input or output commands with this virtual memory scheme. The only thing the user needs to do is reference data. The operating system intervenes to bring the referenced data into main memory if it is not already there. This type of input/output is called implicit I/O. In other words, when implicit input/output is being used the user never issues a read or write command, allowing greatly simplified program structures to be developed. Using

implicit I/O, however, the job's execution is interrupted until the appropriate page has been transferred from mass storage to central memory. For most jobs this is acceptable, especially in multiprogramming mode, because another job can have access to the central processor while the interrupted program is waiting until the implicit I/O is completed. However, when there is one large production type code that is using all the resources of the machine, or a job is to be processed in the shortest possible elapsed time, CYBER 200-OS offers two methods of minimizing program interruptions. The first is the ADVISE request by which the user informs the system, in advance, of the next pages of data which are required, and the system, in parallel with the job execution, streams the data from mass storage to central memory. Then when the job references the data, it is available in central memory and no program interruption takes place. The second is an alternative method of performing input/output called explicit I/O. This is the type of input/output that all programmers are accustomed to, that is, the programmer explicitly uses read and write statements to acquire or store his data. With explicit I/O, a program may be reading, writing and executing simultaneously. This can be important for the effective utilization of the machine in the previously mentioned environments.

With these two file input/output methods, it is possible, based on the user's particular needs, to trade off convenience and efficiency depending on whether the processing time reduction by overlapping input/output and computation for a given job is essential or not.

ACCOUNTING

To provide the elements for the implementation of an appropriate accounting system, CYBER 200-OS generates statistical information about resources used by each task or job step. These statistics, which include a summary of the central processor, central memory and input/output resources used by a job, are appended to the job output as part of the dayfile for that job. All entries are time-stamped and identified by source (user, job, or system operator). Statistics related to the task and its environment are also provided. These include average working set size, number of pages allocated, and page faulting frequency. Additionally, as an installation option, the statistical records may be written to an accounting file maintained and protected by CYBER 200-OS. An installation supplied privileged program is able to extract all accumulated accounting information from the system at any time. The accounting file contains sufficient information to define the scope of a job, so that an accounting program which processes the file can compute a charge by job. At the conclusion of every job, the dayfile information is provided to the user. The information contained in the dayfile includes the resources used, time of day the job started and ended processing, and error messages and informative messages from the operating system and language processors, along with operator actions related to the job.

Accounting and statistical information for the file system is also included. Each file in the system is accounted for by user number, with options for division code and account number.

The following are some file activity statistics which are maintained by the operating system:

- Number of explicit disk accesses and volume of data transferred.
- Number of disk accesses and amount of data transferred due to page faults.
- File creation date.
- File destruction date.

MULTI-TASKING

The operating system supports the distributive processing concept in two ways. First, the operating system itself is divided into small segments which perform specific tasks upon request, and second, the central computer portion of the operating system communicates with the peripheral stations and front-end system(s) using simple commands to effect data and status transfers. The concept is extended to the individual jobs and tasks submitted by users. Each function which the operating system completes is signaled to the user in the form of a standard message.

A job consists of one or more tasks plus the control information. A job starts with the first control card or statement and ends at the end-of-information. Tasks are the execute modules within jobs. Each task can initiate another task. The initiating task, referred to as controller, may pass messages and parameters to the called task, referred to as controllee, and vice versa (see Figure 3-1).

Requests are also interchanged between the user program and the operating system which may in turn generate another request for a disk station or front-end computer to perform, for example, a disk input/output operation.

This mechanism allowing the communication between different user tasks, the operating system's tasks, the front-end, and peripheral tasks, represents an efficient parallelism in program execution, matching the distributed hardware with an equivalent software concept.

JOB PROCESSING CONTROLS

The user is provided with sufficient flexibility to control the way the application is to be processed. The user may keep control over the execution of a specific job or leave it up to the system to advance the job under the normal multiprogramming scheduling constraints. Each batch job processed by the system contains a unique job identification. This allows all output and status information to be correctly associated with the job. As part of the job deck, the user is able to specify resource parameters which include job class, job priority, memory requirements, and total processing time. The user may change the initial memory requirements during each task via job control statements. These resource parameters specified in the job deck are used and enforced by CYBER 200-OS. If either the processing time or memory limits are exceeded during execution, the job will be aborted and an appropriate message returned in the user's dayfile. Users will be allocated real memory up to the amount requested and execution delayed if the requested amount is unavailable. If the memory parameter is not specified, the user's memory space will be dynamically adjusted based upon other activity in the system and the page fault frequency of the job.

CYBER 200-OS allows formal parameters to be specified in stored job control language statements. The user can replace these formal parameters during execution of a job.

Under CYBER 200-OS, the user is also able to obtain control after certain program errors which cause abnormal termination. A user program trying to execute an illegal instruction is an example of this type of error.

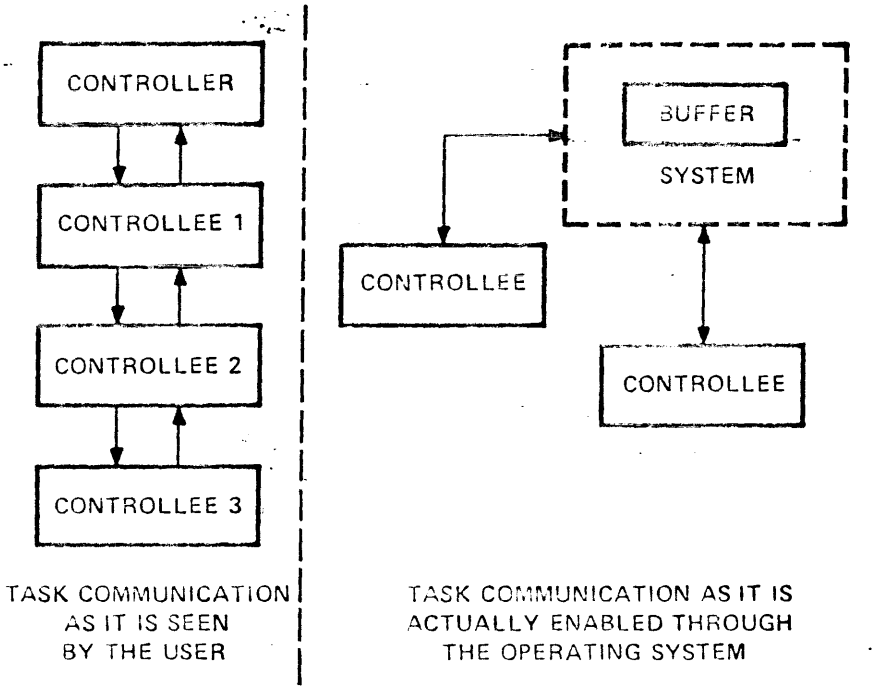


Figure 3-1. Task Communication

The resource parameters associated with user jobs may also be changed through system operator commands. Operator commands are available to change the priority, the amount of virtual memory, program pages and processor time limit of a job. The system operator may also change parameters in the scheduling algorithm such as number of jobs per each priority or job class allowed in execution. These operator commands provide control of job or system resources in order to provide effective processing for all the users of the system.

CYBER 200-OS OVERVIEW

The CYBER 205 operating system (CYBER 200-OS) is divided into three parts. The interactions among the tasks are illustrated in Figure 3-2.

- The small resident system runs in monitor mode; it resides in core and references memory by absolute addresses, rather than through the virtual paging mechanism. When the processor is in monitor mode, interrupts are inhibited and extra instructions are enabled.
- The virtual system tasks, which run in user mode, are called as needed and reference memory by virtual addresses. These tasks communicate with the resident system by using reserved messages and by modifying system tables. They handle non-time-critical tasks such as resource allocation, file management and terminal messages.
- The privileged user tasks have the same characteristics as virtual system tasks but may not modify system tables directly. They perform the physical input/output on devices and pass messages from and to the operator to be acted upon.

Figure 3-3 is a more detailed depiction of the major sections of the operating system and their interaction.

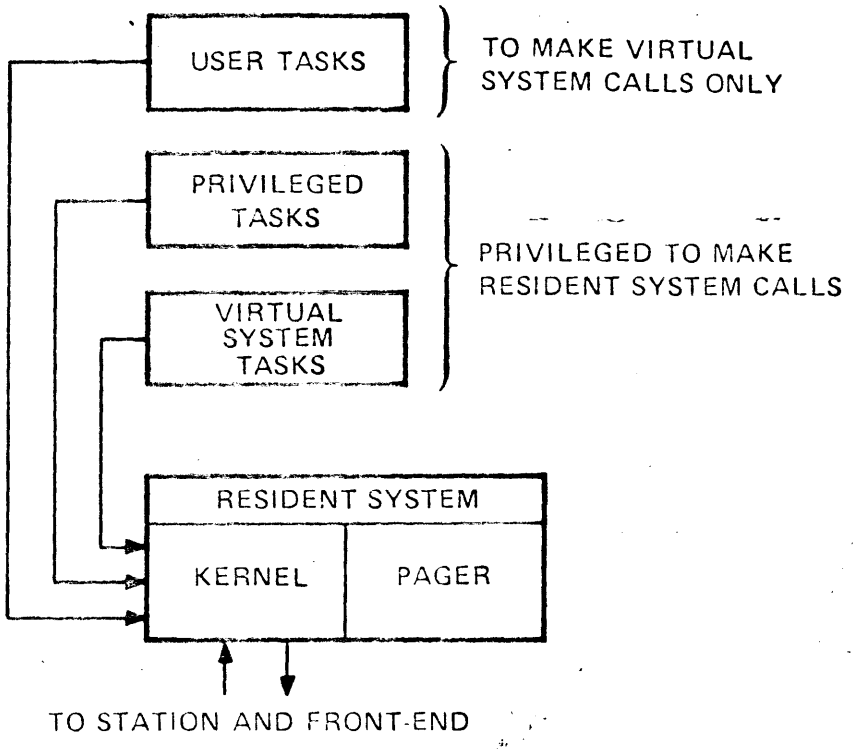


Figure 3-2. Basic System Overview

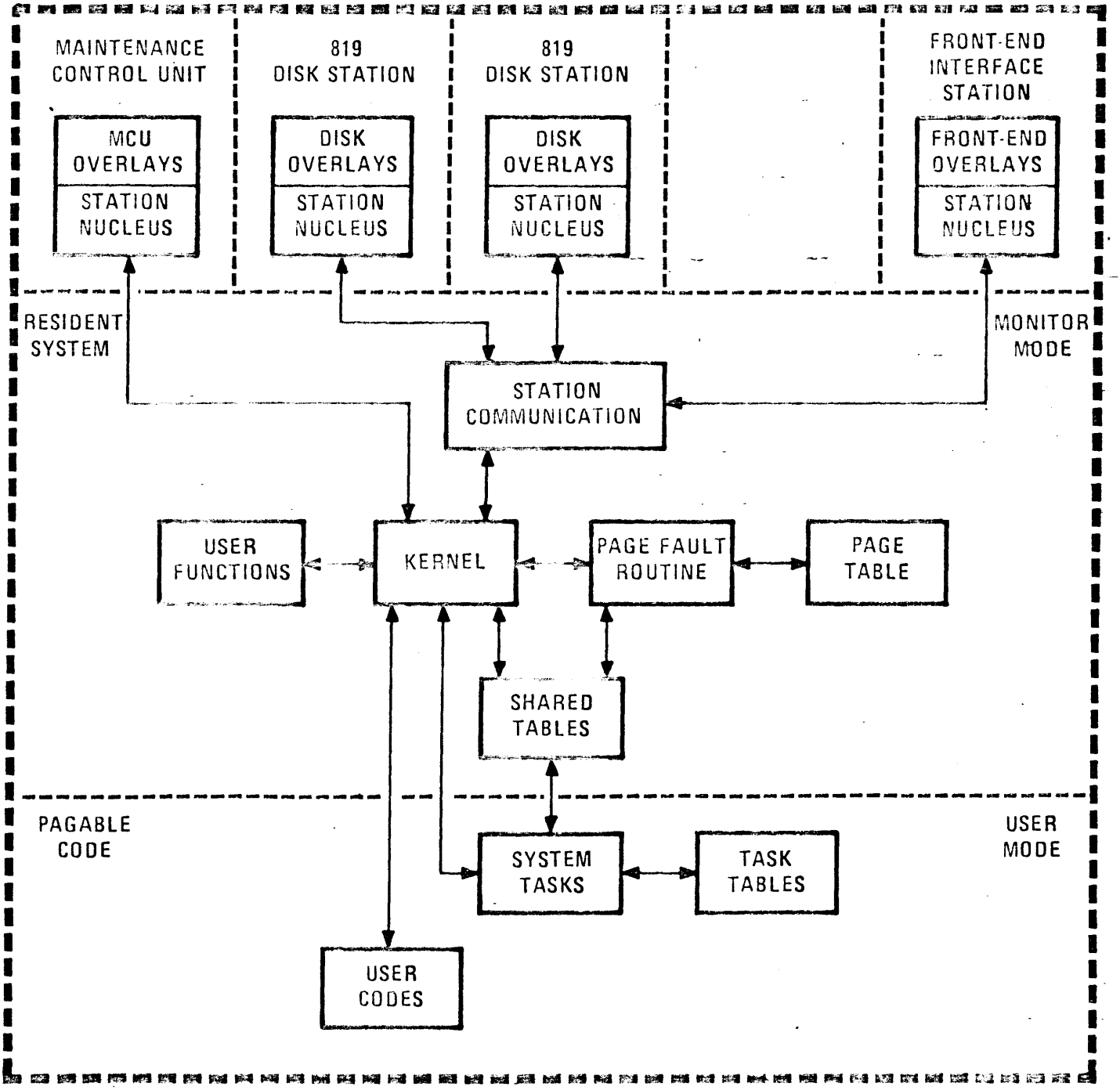


Figure 3-3. Organization and Communication Paths of Operating System

Resident System

The resident portion of the operating system consists of the KERNEL, which handles time slicing of active jobs and message communication, and the PAGER, which handles memory allocation and page swapping.

User jobs, privileged user tasks, and virtual system tasks communicate messages to the KERNEL through a hardware interrupt. PAGER communicates with the KERNEL by setting pointers in the queuing structure without using external interrupts. The KERNEL communicates with the peripheral system by setting pointers and channel flags. All communications between the various portions of the system are by messages. Messages either pass through the KERNEL, in which case it acts as a message switcher, or are processed directly by the KERNEL.

The PAGER dynamically allocates both large and small pages. It performs all required implicit input/output, thereby freeing memory pages and obtaining the required pages from disk storage. PAGER operates in a demand mode. The PAGER routine provides a local paging strategy. Two distinct strategies are used, one for small pages and the other for large pages.

The small page strategy is based on variation of both the common working set and page fault frequency philosophies. A working set is established by noting the distance a marker page migrates downward in the page table over a predetermined time interval. The time interval varies according to the locality and the fault-frequency. Pages which are below the marker page are not part of the working set.

The large page strategy is intimately bound to the envrionment created by the job load and the rules governing the scheduling algorithm for the various priority classes. A job will be allocated a new large page up to its job class maximum, with lower priority jobs being disconnected to make large pages available. Once the maximum is reached, the PAGER will use a "local least recently used" strategy to satisfy large page requests for that program.

The local paging strategy, rather than a "global least recently used eviction" strategy is known as the working set pager; it allows installation definition of applicable parameters.

Associated with paging, the scheduler can control the level of multiprogramming such that page faults are minimized. The scheduler which is tied to the pager maintains four queues for different types of jobs and employs a scheduling algorithm whose behavior may be almost completely controlled by external, locally set parameters. The intent is to provide a given installation with maximum flexibility in scheduling and resource utilization with a minimum amount of page swapping.

Virtual System Tasks

The virtual portion of the system controls the entering of users and jobs into the system, ordering jobs by priority, and entering and removing jobs from the time-slice loop. In addition, it contains routines for system file management, explicit input/output, and terminal message handling.

Virtual system tasks are queued by one of three occurrences:

- Communication from a station requiring processing.
- A user job requesting a system service not provided by the resident system.
- An entry in the periodic table indicating that it is time to run a virtual system task.

During job processing, virtual system tasks connect a user to the system via control cards in batch mode. Virtual system tasks process the messages of the job and perform the end-of-job functions.

Messages from programs and tasks to perform input/output are passed to the virtual system where they are processed and sent to the proper station where the requested input/output takes place. In addition, disk space is allocated through the virtual system, and jobs or tasks are scheduled in to the time-slice mechanism.

Several accounting tasks such as those associated with the file system and system usage accounting are also performed through the virtual system.

All programs share the same virtual system tasks, which are called into memory only upon demand and only to the extent required.

Privileged User Tasks

Privileged user tasks run under special user identification; they can make either normal user calls or privileged system calls and can modify tables only through calls. The system operator under the current CYBER 200-OS is an example of a privileged user.

The system operator monitors and controls interactive and batch tasks running under CYBER 200-OS. The operator communicates interactively with the OPERATOR program to:

- Display user, task, and accounting information
- Terminate tasks
- Suspend and resume tasks
- Enter and display system date and time
- Display virtual system memory or view system tables
- Create and modify user and account information and,
- Logically turn disks and on-line tapes on and off.

Other examples of privileged user tasks are those associated with system file maintenance, and transferring information between a CYBER 200 computer and a front-end computer.

Peripheral System Tasks

Each peripheral station contains a resident portion and a set of overlays for execution of its specified tasks. In the case of a front-end computer, the software is the entire operating system and related software plus a portion of software used for communicating with a CYBER 200 computer system. Communication between CYBER 200-OS and the peripheral stations is by messages and data transfers.

Although related to the recovery and availability of the CDC CYBER 200 computer system, the Maintenance Control Unit (MCU) is part of the peripheral system.

The MCU real-time monitor detects all hardware logic faults, along with temperature, pressure, dew point, and power failures. For catastrophic software failures, the central memory resident processor informs the MCU that a software failure has occurred. Program operation then is halted and control given to the MCU. Station or channel malfunctions are detected by the MCU. This is accomplished by the MCU monitoring the station communication paths, through a series of messages and expected responses, while the system is running.

Failures which require operator attention are reported to the system operator, as well as logged on the MCU mass storage disk. Failures which do not require the operator are simply logged. The error file on the MCU is designed to be dumped at regular intervals and the failures analyzed.

SUMMARY OF OPERATING SYSTEM

The operating system provides the user of a CYBER 200 computer system with numerous features which allow access to the entire range of hardware capabilities in either batch or interactive mode. It manages the distributed processing attributes of the system with a minimal amount of user intervention. Capabilities are provided for user files with several categories of access, defined by the "owner" of the files. Recovery is provided in two forms: one through the operating system if a system hardware or software catastrophic error occurs, and a second, called Checkpoint/Restart, for users.

During operation, statistics and accounting information are gathered and maintained for user and file activity. Provisions are made for a single task to start and communicate with several other user tasks and for jobs and tasks to share files.

LANGUAGE PROCESSORS

CYBER 200 FORTRAN

The source language for CYBER 200 FORTRAN is ANSI (ASA document X3.9-1966) with extensions. The extensions provide vector language syntax and direct access to all central processor instructions which make efficient machine utilization possible without the necessity for assembly language programming. The FORTRAN compiler, object time library and generated object programs are location independent.

The CYBER 200 FORTRAN compiler provides code optimization, loop collapsing into vector instructions and effective utilization of the large CYBER 200 register file. Programmers can write in traditional FORTRAN allowing the compiler to optimize and collapse loops; or, they may utilize the extensions

which permit direct access to the hardware capabilities of vector and string manipulation. Efficient code development is aided by a symbolic cross reference and a symbolic debugging package.

The CYBER 200 FORTRAN library provides the standard and extended mathematical and input/output functions. Although defined primarily for the FORTRAN user, those who are coding in another language may wish to take advantage of the functions in the library. The following sets of routines are included:

- Intrinsic Functions (Scalar and Vector Sets)
  - Absolute Value
  - Truncation
  - Remaindering
  - Choosing Largest or Smallest Value
  - Conversion
  - Difference
  - Precision
  - Complex Factors
- External Functions (Scalar and Vector Sets)
  - Exponential
  - Logarithms
  - Trigonometric
  - Square Root
  - Double Precision Remaindering
  - Modulus
  - System Routines (TIME, DATE, SECOND)
- Input/Output Functions
  - Fixed and Variable Format
  - NAMELIST
  - BUFFER
  - ENCODE/DECODE
  - Unformatted
  - Unit Control (REWIND, BACKSPACE, etc.)

In addition, the Model 205 application user is provided access to the powerful automatic-branching-on-special-conditions facility with the Model 205 computer. This facility is provided for the purpose of detecting and processing unexpected data faults or contingency situations. The user may exercise an

option to provide own code processing of fault conditions or may elect to allow all faults to be handled by the standard error processing routines.

An implementation of ANSI 1977 standards for FORTRAN is currently under development for the CYBER 200 series. However, many of the FORTRAN '77 extensions are provided in the current FORTRAN. Examples of this are the PARAMETER and CHARACTER statements.

CYBER 200 ASSEMBLER

META is the machine language assembler for the Model 205 central processor. The assembler generates relocatable binary output which is linked and loaded by the LOADER under CYBER 200-OS control. META provides:

- Conditional assembly capability for selective assembly.
- Set capability to define, reference and extend the list of expressions.
- Procedure and function capability.
- Attribute assignment for symbols and elements.

CYBER 200 LOADER

LOADER provides the user with a means of collecting and linking relocatable programs and subprograms to produce an executable program. The final product is a file ready for execution under control of the operating system. The loading process involves loading relocatable object modules from one or more user's files and satisfying any unresolved externals from user libraries, if specified, and then from the system library.

LOADER provides many features which control the characteristics of the executable program file. One of the more important features of LOADER is that the user may specify certain routines to be loaded as a group on either a small or large virtual memory page. In a similar fashion, the user may specify data blocks to be grouped on small or large pages, to optimize paging strategies. The virtual memory scheme of the hardware provides each user program with an address space of two trillion words, thus eliminating the traditional use of program overlays, and minimizing memory management constraints on the user.

CYBER 200 SOFTWARE MAINTENANCE AIDS

The source images of the CYBER 200 software system, including compiler and assemblers, are contained in a number of files referred to as program library files. These files, plus other user program and data files, may be created and maintained with a utility program called CYBER 200 UPDATE. The user may choose to maintain these library files using CYBER 200 UPDATE or the CYBER 170 UPDATE on the front-end computer system.

OLE, Object Library Editor, performs several basic functions in the maintenance of the software system. OLE creates CYBER 200 library files by combining files containing binary object modules. It also edits library files by adding or deleting object modules. And, finally it combines files containing object modules producing a non-library formatted file in output.

EDITPUB is the Public File Editor. It allows authorized users to list existing public file names and make additions or deletions to the public files.

The following file utilities are available to all users. These can also be used in the maintenance of the CYBER 200 operating system and library. Control is through the file access permission.

DEFINE	Create a disk file
PURGE	Destroy all or a subset of user private files
GIVE	Give files to another user or pool
FILES	List of user's files along with their attributes
COPY	Copy all or part of one disk file to another
SWITCH	Changes certain file attributes
COMPARE	Compare all or part of one disk file with another
TCOPY	Copy files or records between tapes and disks

These utilities, along with the ones previously mentioned, may be arranged in batch jobs. Using these utilities and by following a well-defined procedure, an authorized user can regenerate the operating system and its product set.

The CYBER 200 operating system which supports these language processors and maintenance aids is a sophisticated system which provides capabilities necessary to control the flow of programs and data in an advanced high-speed computer.

CYBER 200 FRONT-END LINK

The software which interfaces a CYBER 200 through the Distributed Peripheral Network hardware and software to a front-end computer system is called Remote Host Facility (RHF). A front-end system such as the CYBER 170 computer system under control of the Network Operating System (NOS) includes the RHF software. The full set of standard software products provided on both systems are available to the user. This RHF software resides in the CDC CYBER 170 and allows the CYBER to link to another mainframe. Also available is the CYBER 170 series product set which includes FORTRAN, COBOL, ALGOL, data management software, standard utilities, etc., as well as diagnostics. The Model 205 runs the standard CYBER 200-OS as well as another portion of the RHF software, the complement of the CYBER 170 software. On the Model 205, there is also a full set of diagnostics which run either on-line or off-line.

One of the benefits gained using RHF software is a multi-mainframe environment. If desired, the user has a choice of different mainframes. A standard large-scale computer such as the CYBER 170 series works very well at editing and job preparations by doing this work on the front-end processor; the Model 205 is free for large computational tasks and, therefore, better utilization is made of the Model 205 computer. Another advantage is that one set of peripherals can serve the needs of both the Model 205 and the front-end system. All the slower peripherals, tape drives, card readers, printers, etc., can be on-line to the front-end system, but the Model 205 still has access to these peripheral devices via RHF.

Note that high-speed random access storage devices, the CDC 819 High Capacity Disks, and the CDC 679 high-speed tapes, are on-line to the Model 205. Using CDC-supplied software for the front-end system and the Model 205 system allows use of the mature CDC CYBER 170 software as well as access to the new software and new features of the Model 205 computer.

RHF offers spooling, staging and interactive capabilities. Spooling is defined as a local or remote job file being submitted to the front-end system input queue for execution on the Model 205. A parameter on the job card specifies the destination mainframe in this case and the job file is transferred to that mainframe for execution. Output generated during execution of the job is automatically returned to the originating mainframe and terminal unless explicitly diverted by the job.

The staging capability allows the files to be transferred from the front-end system to the Model 205 and from the Model 205 to the front-end system. The Model 205 user can request via control card that a mass storage file located on the front-end system be staged to the Model 205 or that a mass storage file located on the Model 205 be staged to the front-end system.

One of the considerations in data transmission between the Model 205 and the system front-end is file conversion. A job file is transferred over to Model 205 and converted into a sequential structured file, recognizable by the Model 205 batch processor. When the output is transported back to the front-end system, it is then converted to a printable or punchable file. A coded file is converted on the Model 205 to an ASCII file. A front-end system binary file located either on mass storage or tape is transferred with or without conversion to the Model 205 mass storage. With conversion, the binary files are converted from front-end system format to Model 205 64-bit format. In a similar fashion, data conversion is performed in a reverse manner when the file is transferred from the Model 205 to the front-end system.

The Model 205 software can also be accessed interactively at a front-end system terminal. The interactive messages input at the terminal are transferred to the Model 205 system across the link, completely transparent to the front-end system. In addition, the input, print, punch, and active Model 205 queues can be displayed on a terminal.

The Model 205 interactive user has all the same Job Control Language and other capabilities as the Model 205 batch user. In addition, the Model 205 interactive user has available utilities for testing and debugging correctly compiled programs that execute unsatisfactorily. For example, the DEBUG utility allows a user to breakpoint at a place in the program, display and then alter data in the program, and finally continue execution from the last user breakpoint. This process of breakpointing, displaying data, and then continuing execution can be used to speed the calendar time associated with obtaining a correctly executing program. DEBUG interfaces to the symbol table generated by FORTRAN. Other utilities, such as LOOK which display or alter the contents of mass storage files, are also available to help the user in developing new programs in an interactive mode.

Production programs, as well, can be run in interactive mode. A user may wish to observe the effects of parameter choices on program execution. Parameters can be entered at a terminal, and significant data displayed, allowing the user to control the processing.

The RHF software design is such that standard utilities exist internal to itself to handle all file, commands, and data communication requirements. These utilities are the small virtual tasks which handle the specialized requirements, while the resident tasks handle the generalized or global requirements. The resident task runs the communication buffers and the utility tasks handle data conversion, file read and write requests, message formatting and parameter gathering.

The design of the RHF software permits multiple front-end computers to be attached and in operation concurrently. This multiple front-end configuration to the Model 205 can grow, limited only by the number of front-end systems devoted to computer linkage.