PRIME Product Bulletin

PRIMOS IV and V

THE PRIME OPERATING SYSTEM

All Prime computer systems—from small, dedicated systems using the Prime 100 central processor, to large virtual memory Prime 500's that support dozens of concurrent real-time, time-shared and queued tasks—use a common, uniform operating system called PRIMOS. Since each central processor provides a different level of performance within a common hardware architecture, PRIMOS is implemented in distinct but compatible levels to maximize the effectiveness of a processor's resources while minimizing operating system overhead.

As illustrated, PRIMOS is currently offered on four levels: PRIMOS II, III, IV, and V. PRIMOS II provides an interactive, single-user, disk operating system, and a modular, memory- or disk-resident, multi-task, real-time operating system for Prime 100 and 200 central processors.

PRIMOS III is geared to the paged memory management system of the Prime 300. It provides a virtual memory disk operating system that can support 31 simultaneous users and a real-time operating system that can operate independently or in a fully protected foreground-background mode with a single-user disk operating system in the background.

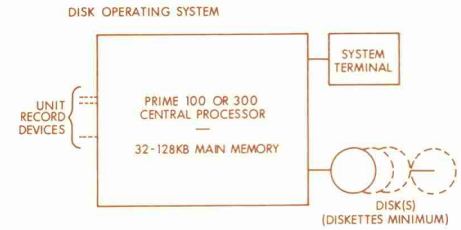
PRIMOS IV optimizes the Prime 400's high-speed computational ability and exceptionally large memory capacity by integrating interactive, queued-job and real-time supervisory services into a single "embedded" operating system. The Prime 400 features paged and segmented virtual memory with an address space of 0.5 billion bytes, an interleaving main memory system of up to eight million bytes with a 2K-byte bipolar cache memory, and a disk capacity that can exceed 2.4 billion bytes.

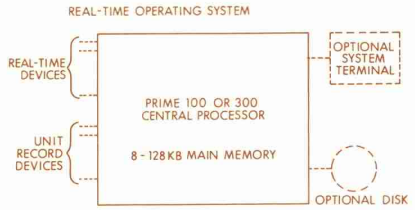
PRIMOS V, available with the top-of-the-line Prime 500, includes all the features and capabilities of PRIMOS IV plus assembly language support for the Prime 500's general register, 32-bit architecture. Under PRIMOS V, decimal arithmetic, character manipulation, and character editing instructions are directly executed with a combination of hardware and firmware. On the Prime 400, these instructions are automatically trapped to software subroutines at run-time.

512 Million Byte Virtual Memory Ends Program Size Limitations

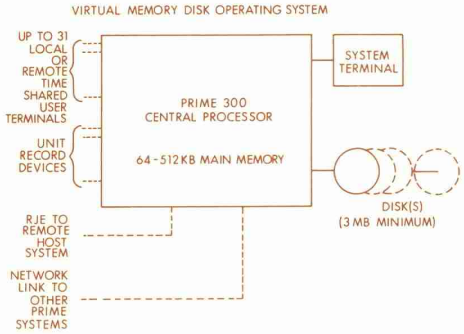
PRIMOS IV and V's virtual memory management systems (a combination of paging and segmentation) make it possible for both the Prime 400 and 500 to run multiple processes each within a private 256 million byte virtual memory space, plus an additional 256 million byte space that is automatically shared with all other processes. PRIMOS IV and V can run as many as 63 interactive time-shared users at local or remote

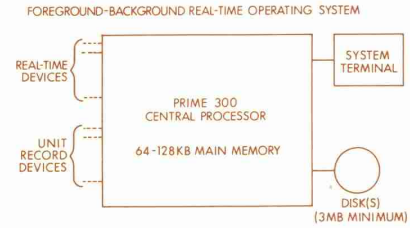
PRIMOS II



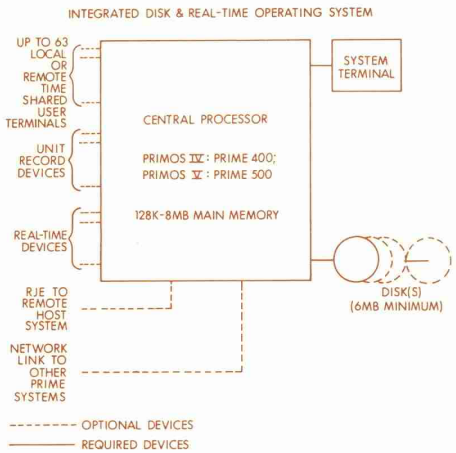


PRIMOS III





PRIMOS IV and PRIMOS V



PRIMOS HARDWARE REQUIREMENTS: MINIMUM CONFIGURATIONS FOR PRIMOS II, III, IV, V

terminals. Furthermore, virtual memory resources are available on systems with as few as 128K bytes of main memory. PRIMOS IV and V can automatically take advantage of additional increments of main memory which can be added (up to the maximum capacity of eight million bytes) to minimize the paging demands of an expanded virtual memory. This feature maintains a consistently high level of system responsiveness as the number of processes controlled by the system increases.

PRIMOS TERMINOLOGY

FILE—a named collection of data or code that is created, manipulated, or deleted by Prime File Management System.

PHANTOM USER—a time-shared process that requires no interaction with a user. Once initiated, it runs to completion without further user interaction.

PHYSICAL ADDRESS SPACE—an array of real or main memory space addressed by word number (0-4,194,303).

PROCEDURE—a subroutine referenced by the Prime 400 Procedure Call mechanism.

PROCESS—a program running in a virtual address space.

QUEUED PROCESS—a process that waits with other such processes to run in a first-in first-out (FIFO) sequence. Only one process at a time runs as an active time-shared process. A second time-shared process, the queue manager, controls the queue.

RE-ENTRANT PROCEDURE—a procedure that can be called by several concurrently executing processes.

SEGMENT—a 128K-byte portion of virtual address space. Access rights are assigned on a per segment basis. Calculations of segment number are usually made by PRIMOS IV and V; a process makes named references to procedures and data arrays.

SHARED PROCEDURE—a re-entrant procedure that is present only once in physical memory regardless of how many processes are concurrently executing it.

TIME-SHARED PROCESS—a process that gains use of the central processor for a fixed time period in cyclical order with other such processes.

USER—a time-shared process or a person associated with an interactive terminal.

VIRTUAL ADDRESS SPACE—an array of 512 million bytes (or 256 million words, as 1 word = 2 bytes = 16 bits) that a process references by segment number (0-4,095) and word number (0-65,535). Half of the virtual address space is unique or private to each process, while the other half is common to or shared by all processes.

VIRTUAL MEMORY—that quantity of address space stored in disk form. Data stored in virtual memory is transfered to real or main memory as needed by the executing process.

Embedded Design Gives 20-Times Faster Access

PRIMOS IV and V are exceptionally responsive operating systems because they depart from traditional designs in which the operating system was a separate, self-contained body of software that operated by remote control through intricate and time-consuming sequences of interrupts and responses. Instead, PRIMOS IV and V provide direct and immediate control because they are embedded in the virtual address spaces of all processes using the system. In fact, PRIMOS IV and V are an integral part of each process. Each operating system responds immediately to all commands from users of terminals, program calls to the file system, library routines, and other shared procedures. Users are able to access any operating system resource in no more time than it takes for a user program to call a sub-routine, which is about 1/20 the access time of conventional operating systems. Note that while operating system and processes share common address spaces, a comprehensive, multi-ring protection system completely protects the operating system against improper access or accidental modification.

Upward and Downward Compatibility Protects Your Software Investment

Programs and data files created on one Prime computer can be used on any other larger or smaller Prime computer without modification. This compatibility among all systems holds true not only at the source language level but at the memory-image level as well. Thus it is possible, for example, to create a memory-image file on a Prime 200 controlled by PRIMOS II, and run the file directly on a Prime 400 controlled by PRIMOS IV or a Prime 500 controlled by PRIMOS V. Furthermore, FORTRAN, COBOL, and PMA programs that ran on a Prime 300 can be recompiled to run on a Prime 400 or 500 and utilize their expanded capabilities. More significantly, the reverse is also possible.* Thus a multi-user Prime 400 or 500 can be used as an extremely efficient and economical software development system, creating programs and databases that can be transferred directly to smaller dedicated application systems using Prime 100 or 200 central processors.

There are many direct benefits of such upward and downward compatibility. Obviously, a user's programming investment is preserved when upgrading to a larger system. The development of software and the communication among systems in a network of Prime computers is streamlined. Additionally, it encourages the establishment of programming and system design standards that are completely transferable among all Prime products.

*Of course, programs developed for smaller systems must anticipate the limited hardware resources of such systems.

Comprehensive System Integrity and Security Features Offer Complete Protection

A combination of hardware, firmware (microprogrammed logic), and software components within PRIMOS IV and V monitors the complete hardware/software system to assure the user that the hardware is operating reliably and that processes being executed are secure.

System integrity is constantly monitored by features such as main memory error correcting codes (an optional feature that automatically detects and corrects all single-bit memory errors) and microverification (microprogrammed routines that test the central processor's logic and help determine the cause of faulty operation). PRIMOS IV and V allow the system operator to "lock out" 2K-byte pages of main memory should errors be detected and reported within the pages. The operating systems also include file access integrity features such as forward and backward pointers and built-in utilities to regenerate record availability tables.

PRIMOS IV and V complement these integrity features with a multi-level security system for all users, programs, files and the operating system itself. Certain security measures, such as the protection of the operating system from users and other processes, are automatic and unalterable by any user or program. Other measures, such as the permission scheme necessary to access files, can be tailored by the user to match the needs of each installation.

Prime's Philosophy is: Software First!

PRIMOS was designed before the computers it operates. This unique software-first design philosophy is the key to Prime's ability to offer a range of processor performance and software functionality unmatched on systems costing ten times more. Software first explains why Prime alone offers upward and downward program compatibility among all central processors. Software-first made it possible to establish a uniform file system used by all levels of PRIMOS, all language translators, all utilities and all libraries. Software-first is why Prime systems are so efficient: hardware features have been designed specifically to optimize software performance. In short, software-first is the difference between buying a computer with some software included, or buying a completely integrated software/hardware system—from Prime.

WHAT PRIMOS IV AND V DO ...Without Being Asked

One of the key factors in evaluating an operating system is not how it responds to user-initiated commands, but what it does for the user automatically, without being asked. These are the features that determine how flexible and efficient the system is in managing basic resources such as memory and disk file space, CPU time, I/O devices, data communication, etc. These are the implicit operating system functions that, if properly designed and implemented, are totally transparent to the user. Such transparancy means major time savings during interactive program development and at job run time.

Implicit Functions

Time Scheduling. PRIMOS IV and V regulate the amount of central processor time available to active time-shared processes by assigning time slices. A time slice, which is normally set to a 1/3 second duration, represents the maximum continuous time that a time-shared process may retain control of the central processor. Time slices are allocated on a priority basis with interactive processes receiving a higher priority and queued processes a lower priority.

Memory Management. Associated with each process is a 512 million byte virtual address space. This address space is organized, or mapped, as follows: 4,096 segments, each segment containing 64, 2K-byte pages. The memory management system tracks the location of all pages and handles the physical movement, or swapping, of pages between disk and main memory. Thus, the memory management system automatically transfers the page from the disk to the least-recently-used area of main memory.

Procedure Sharing. To hold page turning to a minimum and thereby maximize system response time, any number of processes can share the same copy (or page) of a common procedure once it is in main memory. For example, in a multiuser environment, all users can share one copy of the Editor instead of each user having to page-in a separate copy. Shared procedures are reentrant, so they remain unaltered by the processes that use them.

Security. A combination of PRIMOS software and Prime 400/500 hardware protects the operating system from the processes using its services, protects processes from each other, and enforces the access privileges established for all shared procedures and files. The security system automatically checks the validity of service requests such as read, write, and execute, and prevents such services from being performed if pre-established access rules have not been satisfied by these requests.

Disk Space Allocation. PRIMOS IV and V automatically assign logical files to physical disk records. This feature permits a user to create file structures without concern for the type of disk that stores the files or its operating format and physical characteristics. The operating system provides forward and backward pointers to every disk record to ensure file integrity and simplify access to the next record in a file.

I/O Handling. Built-in to PRIMOS IV and V are standardized resources for servicing interrupts, maintaining status information, and controlling data transfers for all Prime peripheral devices. These resources permit a user to communicate with any peripheral device using high-level call statements after the user's process has either been attached to a device (e.g., magnetic tape or printer) or to the files contained on a disk device. A device such as a printer is shared among several users by submitting files to a spool manager that queues the files on a disk and schedules printing on a first-in, first-out basis.

Data Communication. PRIMOS IV and V handle all data communication between a Prime 400 or 500 system, a wide variety of interactive terminals, other Prime central processors and several EDP mainframes. PRIMOS IV and V communicate directly with most currently available asynchronous ASCII terminals operating at speeds up to 9600 baud. Communication with other Prime central processors is handled via interprocessor controllers for locally connected processors, or via high-speed synchronous lines using a standard packet-switching host-access protocol (an initial implementation of the CCITT X.25 international standard). Remote Job Entry (RJE) communications are supported by IBM 2780, IBM HASP, and CDC UT-200 protocols.

WHAT PRIMOS IV AND V DO ...When Asked

The part of PRIMOS IV and V that is immediately visible to a user contains the resources that respond either to explicit user commands or procedure calls. These resources provide economical and easily accessible services to all users in the amount they need, without advance notification or the intervention of a central operator. Additionally, an extensive list of commands and calls can be used without a broad knowledge of the inner workings of the operating system or central processor. The major services provided by these resources are described below.

Job Control

PRIMOS IV and V schedule up to 63 time-shared processes including interactive users, phantom users, queued processes, and RJE processes. Additionally, PRIMOS IV and V can schedule as many as several hundred real-time processes.

Interactive Users. As many as 63 interactive terminals can be on-line concurrently. The system handles any mix of direct-connected local terminals and modem-interfaced remote terminals. Individual users have complete freedom to use system resources as if they were its sole users. The user issues commands from a terminal keyboard and gains immediate access to PRIMOS IV and V's resources, which include language translators, editors, debugging aids, and a variety of file structures. Immediate interaction between the user and the user's job is ideally suited for program development. The user controls the job and handles exceptional conditions as they occur instead of depending on a batch processor, thus completing the job in minutes instead of hours or even days. Interaction is what PRIMOS IV and V are all about.

Phantom Users. A phantom user is a process that, once initiated from a terminal, requires no further user interaction or terminal output until it is completed. The user can continue to initiate other phantom user processes from the same terminal, use the terminal interactively, or log it off the system. The operating system treats phantom and interactive users the same way, except that interactive users are given a higher priority for time slices.

Queued Jobs. Some jobs, like disk-to-tape media conversion, sorting, and report printing have turnaround requirements that are less stringent than those of interactive and phantom users. Such processes can be collected in a job queue and run on a first-in, first-out (FIFO) basis under control of a job queue manager. Executing the processes sequentially rather than concurrently conserves time slices so the system can maintain a high level of response to interactive processes.

Remote Job Entry. A Prime 300, 400, or 500 can act as an RJE system by emulating the protocols used by the IBM 2780, IBM HASP, and CDC UT-200. When any is used as an RJE system, card equipment can frequently be eliminated since programs and data files can be created interactively and then queued on a disk for direct transmission to a host mainframe.

Program Development

PRIMOS IV and V provide an extremely efficient environment for interactive program development. Not only do they assure fast response to all users, they also offer each user access to a complete set of sophisticated software development tools including state-of-the-art COBOL, BASIC and FORTRAN compilers, RPG II, text editors, and utilities.

COBOL. Prime's COBOL is an implementation of the 1974 ANSI standard. By adhering to this standard, Prime provides an economical migration path for current program libraries to be transferred to the Prime 400 or 500 from other systems with minimal conversion.

RPG II. Prime provides an RPG II translator that is functionally comparable to the one used with IBM's System 3 Model 10. However, when run under control of PRIMOS IV or V, RPG II applications can easily be expanded, using languages such as COBOL and FORTRAN IV and the common file system to exploit PRIMOS IV and V's interactive and time-shared capabilities.

FORTRAN IV. Prime's FORTRAN is 1966 ANSI-compatible FORTRAN IV with extensions. It is processed by a very efficient one-pass compiler that produces highly optimized code which rivals the output of most macro assemblers. As an indication of the flexibility offered by Prime FORTRAN IV, note that FORTRAN IV is the major systems programming language used by Prime. In fact, over 80% of PRIMOS IV and V is written in FORTRAN.

BASIC. Prime's BASIC supports three modes of operation—conversational, queued, and immediate to satisfy the computational needs of a wide range of users. BASIC shares the same file system used by all other Prime language processors, and programs written in BASIC can call subroutines written in FORTRAN, COBOL, or assembly language.

Macro Assembler. The Prime Macro Assembler (PMA) is a free-form, symbolic programming system providing extensive macro facilities that simplify the creation and usage of application-oriented commands. PMA includes over 60 pseudo operations for such functions as assembly control, listing and loader control, variable definitions and storage allocation, program linking, and addressing mode control.

File Management

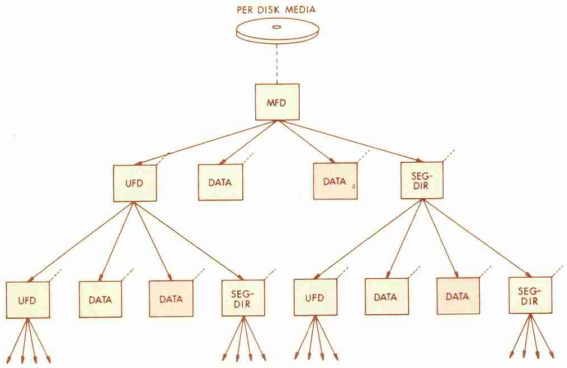
The Prime File Management System provides both implicit and explicit file management resources. The implicit, or automatic, resources manage the allocation of named files to physical address spaces in main and disk memories. The explicit resources provide the file access methods and interactive tools necessary to add, modify, and delete files. PRIMOS IV and V support the same file structure as PRIMOS II and III. As a result, program files and data bases created under one level of PRIMOS are directly transferable to any other level, where they can be used without modification.

The full capabilities of the file system are available to COBOL, FORTRAN, and PMA and can be used to develop special file structures and access methods. However, most users simply refer to files by name which are then constructed and located by procedures such as the editor, language translators, and keyed index/direct access (KI/DA) method.

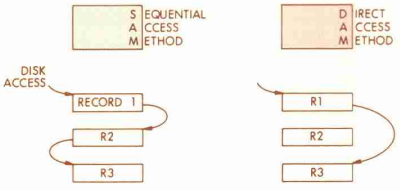
The file structure, as illustrated below, can be viewed as a hierarchy or tree-structure containing specialized file directories and data files.

Master File Directory. The master file directory (MFD) is the highest level in the file structure. The file system creates and maintains an MFD for a disk or a user-specified portion of a disk. The MFD contains the names and locations of user-file directories, segment directories, and data files.

PRIMOS IV FILE STRUCTURE



ACCESS TO RECORDS WITHIN A FILE



User File Directory. User File Directories (UFD's) are usually associated with individual users and user processes, and contain pointers to named data files and additional file directories.

Segment Directory. A segment directory (SEG-DIR) is a file that contains pointers to subsidiary files. It permits rapid access to large collections of data that have an established order but variable size. Files are accessed by simply indexing by pointer position within a SEG-DIR.

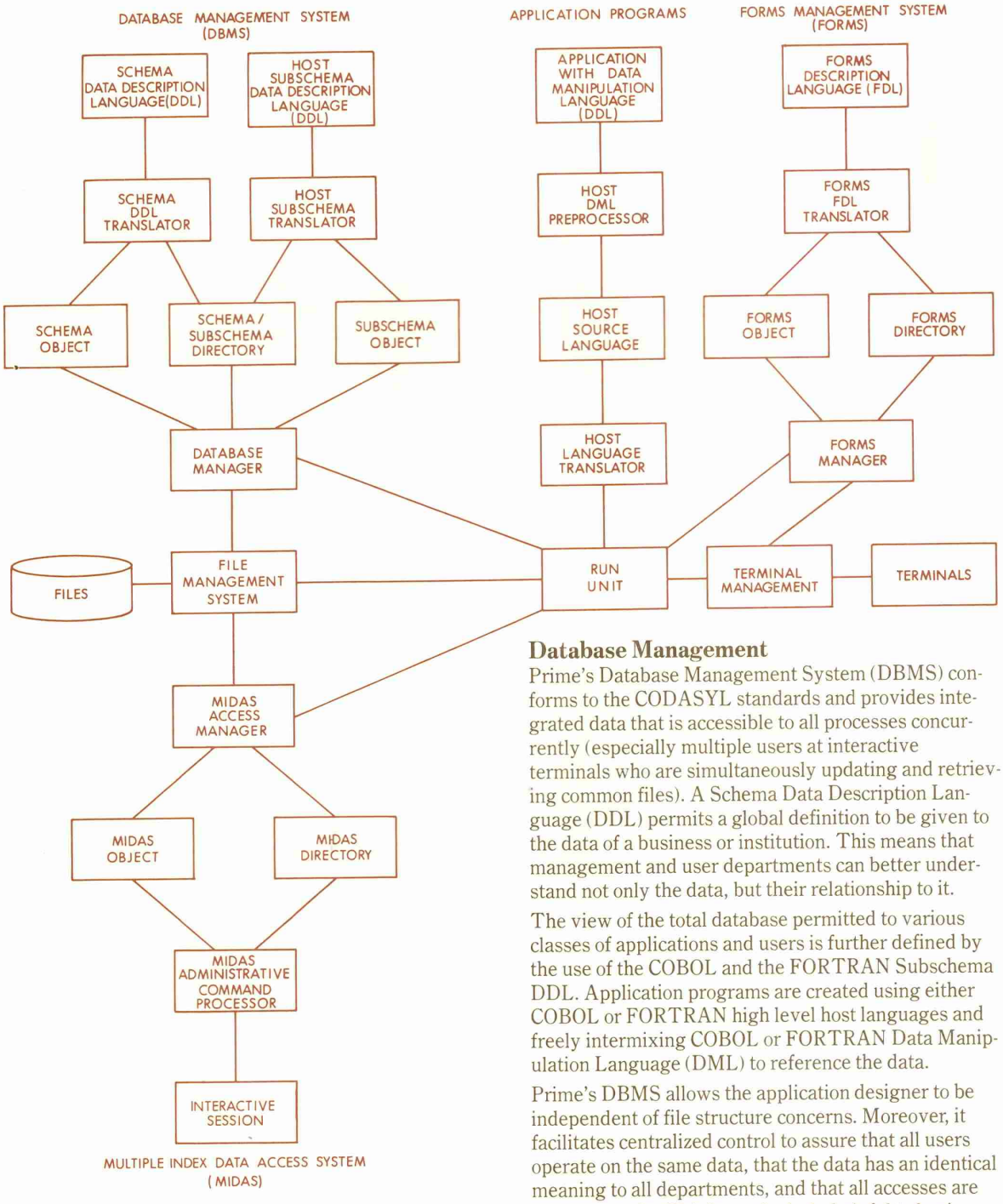
Sequential and Direct Access Methods. Files can be constructed to permit access to their contents using either a direct access method (DAM) or sequential access method (SAM). By convention, all file directories used by the operating system are structured as SAM files. In SAM files each record contains a pointer to the next record in sequence, thereby reducing the number of directory accesses needed to search the file. DAM files, in contrast, store pointers to all records in the first record of a file. SAM or DAM files may be of any size that can be accommodated on the disk.

Keyed Index/Direct Access Method. KI/DA provides a fast and versatile method for locating, adding, deleting, and modifying items in any size data file. Up to 20 different key fields can be used to access a single item. KI/DA's shareable procedures use the SEG-DIR and DAM capabilities of the Prime file management system to minimize the time required to locate an item.

Data Management

Prime's Multiple Index Data Access System (MIDAS) is a part of PRIMOS. It bridges the gap between the File Management System (FMS) and the Database Management System (DBMS). MIDAS files are directly supportable under DBMS; this permits a controlled migration from MIDAS to DBMS. MIDAS uses the resources of FMS and, in particular, relies on KI/DA as its access method. When combined with Prime's Forms Management System (FORMS), MIDAS becomes the basis for the development of transaction-oriented systems. MIDAS permits many users to access fixed- or variable-length records with locks specified at the data record to avoid concurrent usage conflicts. A single program can sequentially and randomly access a MIDAS file and do complete or partial file searches based on any combination of up to 20 keys with duplicates. Furthermore, each key can contain data in any format (e.g., single- or double-precision floating point, integer, ASCII, etc.) so conversion to a common format is unnecessary. MIDAS interacts with users and, by a series of questions and answers, describes, creates, maintains, and manipulates large, structured data files. MIDAS files are available via CALLs and standard READ/WRITE statements to application programs written in any of Prime's languages on 300, 400, and 500 central processors with at least 128K bytes of main memory.

APPLICATION IMPLIMENTER'S VIEW OF
THE INTERACTIVE COBOL DATA PROCESSING ENVIRONMENT



Database Management

Prime's Database Management System (DBMS) conforms to the CODASYL standards and provides integrated data that is accessible to all processes concurrently (especially multiple users at interactive terminals who are simultaneously updating and retrieving common files). A Schema Data Description Language (DDL) permits a global definition to be given to the data of a business or institution. This means that management and user departments can better understand not only the data, but their relationship to it.

The view of the total database permitted to various classes of applications and users is further defined by the use of the COBOL and the FORTRAN Subschema DDL. Application programs are created using either COBOL or FORTRAN high level host languages and freely intermixing COBOL or FORTRAN Data Manipulation Language (DML) to reference the data.

Prime's DBMS allows the application designer to be independent of file structure concerns. Moreover, it facilitates centralized control to assure that all users operate on the same data, that the data has an identical meaning to all departments, and that all accesses are made securely. This is particularly helpful to businesses and institutions with complex data relationships and changing information requirements.

Prime's DBMS means reduced application programming expenses and shorter development times. The programmers concentrate on the logic of the application, not the details of data manipulation and file design, and sort/merge operations are significantly reduced.

Forms Management

Prime's Forms Management System (FORMS) is a set of software functions that are used to develop systems for interactive, multi-terminal, transaction processing. FORMS permits forms to be designed for a variety of CRT and hardcopy terminals using the Forms Description Language (FDL) with easy-to-use statements. Application programs are created using COBOL, FORTRAN, or PMA with standard Read/Write statements.

FORMS may be used separately or in conjunction with the DBMS. The application program is unbound from the form's description, the type of terminal, and the data-base description until the program is run. This independence is similar in philosophy to the DBMS' facilities and results in similar savings in programming expense and time to develop new applications initially and in response to changing requirements.

Networking

The reach of the Prime File Management System can be extended to include files contained in other Prime 300, 400, and 500 computers. This is accomplished automatically by PRIMENET, Prime's network software. PRIMENET makes it possible for a user or process on one Prime computer to access files on any other Prime computer in a network, without concern for any of the protocol details involved in managing the data transfer. Thus, regardless of where a file is actually located, PRIMENET makes it appear to the user that it is within the system he is connected to. In addition, PRIMENET supports a wide range of network activities:

- A user can, with proper password identification, log in to any computer in a network from any terminal in the network.
- Users can run their programs on a remote system by logging into that system.
- Similarly, printing jobs can be spooled on a remote system.
- Processes running concurrently on different systems can communicate interactively with one another via special transmit and receive calls.
- An operator at a system terminal can send messages to a user on the local system or to all on-line network users.

HOW PRIMOS IV AND V WORK

PRIMOS, in conjunction with key hardware features in the Prime 400 and 500 central processors, performs four major functions: activity scheduling, memory management, procedure sharing, and system protection.

Activity Scheduling

PRIMOS IV and V automatically transfer the attention of the central processor from one activity (or process) to another with minimum overhead and complete protection. The key is a central processor feature called Process Exchange. As illustrated, this feature is a hardware dispatcher that manages the Ready List, a number of Wait Lists, Semaphores, and the Process Control Blocks containing detailed control and status information for each process.

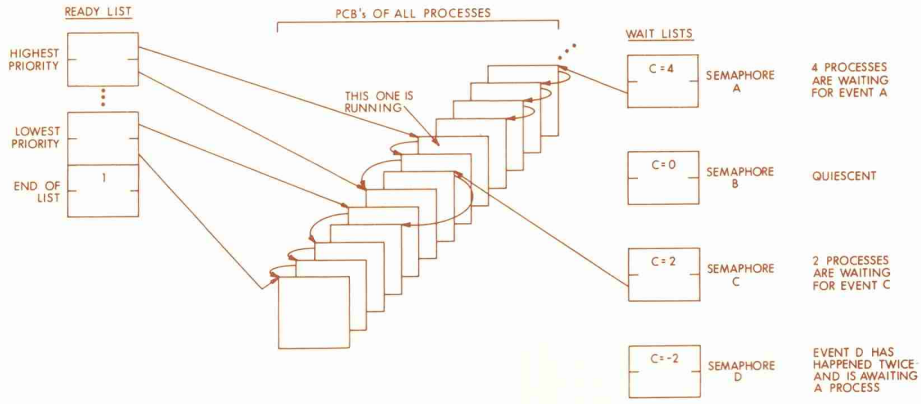
Process Exchange. Exchanges are caused asynchronously by hardware-generated interrupts, faults and checks, and synchronously by a process executing WAIT and NOTIFY instructions. These events activate the dispatcher to re-order the lists and get the highest priority process to run. This exchange takes 7 to 50 microseconds; without the hardware assist it would take 200 to 500 microseconds. The dispatcher also manages the processor's live registers, permitting sets of registers to be assigned to different processes. This reduces the need to save and restore the registers' contents and speeds the process exchange.

Ready List. The Ready List identifies all processes that are ready to run. The list is ordered first by priorities and then chronologically. The highest priority process is the one running, the others are run in order of their priority as they appear on the Ready List.

PROCESS EXCHANGE EXAMPLE

EVENT	READY LIST	SEMAPHORE 1 (WORK FOR B TO DO) WAIT LIST	SEMAPHORE 1 (WORK FOR B TO DO) COUNT	SEMAPHORE 2 (B HAS DONE IT) WAIT LIST	SEMAPHORE 2 (B HAS DONE IT) COUNT
A STARTS	PCB A	PCB B	+1	–	0
(A RUNS α)					
A NOTIFIES S1	A, PCB B	–	0	–	0
(B RUNS δ)					
B NOTIFIES S2	A, PCB B	–	0	–	-1
B WAITS FOR S1	PCB A	PCB B	+1	–	-1
(A RUNS β)					
A WAITS FOR S2	PCB A	PCB B	+1	–	0
(A RUNS γ)					

ACTIVITY SCHEDULING



Semaphores and Wait Lists. Each event that can cause an exchange is associated with a Semaphore (two words in memory) that keeps a negative count of the number of times the event has occurred without being serviced by a process, or a positive count of, and a pointer to, the processes awaiting the event (there is a Wait List for each Semaphore). The processes on a Semaphore's Wait List are ordered and serviced similar to the Ready List. Semaphores are associated with such events as: 'wait for a 1/3 second time slice,' 'disk read complete,' and 'character received from a terminal.'

To schedule a time-shared process, for example, an interrupt from the Real Time Clock causes the hardware to notify the waiting time-keeping process, which awakens and exchanges run status with the previously running time-shared process. Once the time-keeping process begins running, it updates several counters including the 1/3-second counter. If the 1/3-second counter is full, indicating that 1/3-second has transpired since it was last reset, the time-shared process of next highest priority exchanges its ready status with the previous time-shared process through a series of WAIT and NOTIFY instructions. The time-keeping process now waits for the next Real Time Clock interrupt and the highest priority process awakens and starts to run.

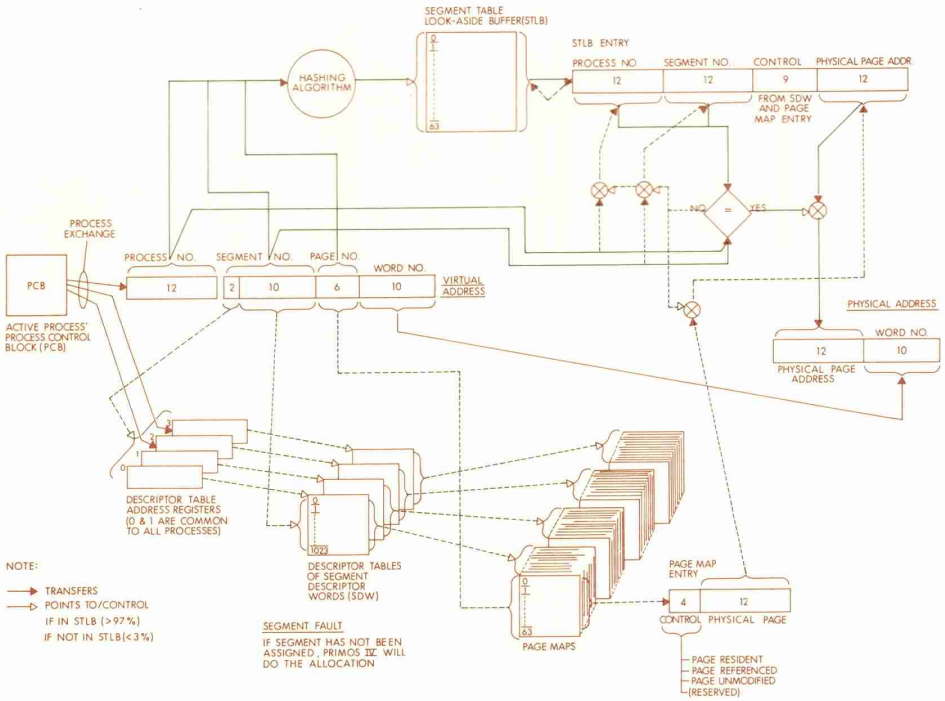
Memory Management

The user writes programs in any of Prime's languages, and is unaware of the memory management scheme unless a subroutine or an array exceeds 128KB (one segment). If the program occupies one segment or less, the user may instruct the language translators to generate memory reference code that is compatible with a smaller Prime central processor. For larger programs, the user makes symbolic reference to subroutines and data arrays. The translators and loaders establish a correspondence between the symbolic reference and a virtual address.

Virtual-To-Physical Address Translation. The virtual address contains a segment number (one of 4,096), a page number (64 2K-byte pages/segment), and a word number (0 to 1,023; one 16 bit word = two bytes). The virtual address is translated into a physical address by a series of segment tables and page maps stored in main memory (see illustration). To speed translation, a Segment Table Lookaside Buffer (STLB) is used. For a typical mix of operations, the required address information will be in the STLB more than 97% of the time and no overhead is added to the memory cycle time.

Demand Paging. When a process references a page that is not in main memory, a "missing page" fault occurs and is serviced by PRIMOS IV and V. PRIMOS IV and V then replace the least-recently-used (LRU) page with the referenced page. The next 'n' LRU pages are also freed to satisfy subsequent demands. If they have been modified, they are first written back to disk before their memory space is overwritten. This demand page algorithm has been designed to reduce the number of disk accesses.

AUTOMATIC MEMORY MANAGEMENT SYSTEM



Cache Memory. Two features of the central processor speed the effective memory cycle time: cache memory and interleaving. Cache memory is a high-speed bipolar, 2,048 byte memory, that retains the contents of the most recent memory references and is addressed on the basis of word number (the least-significant 10 bits of a memory address). When referencing memory, the cache is read first while the STLB is checked for correspondence. If correspondence exists, the word read from the cache is used; if not, the correspondence is made and the word is read from main memory and stored in the cache. When a word is written in main memory, it is also written in the cache. On average, more than 85% of memory references are found in the cache, reducing the effective memory access time. The access time is 80 ns from the cache and 600 ns from main memory.

Memory Interleaving. The processor interleaves memory references to pairs of sequentially addressed memory words. For example, if word 6 or 7 is read, both 6 and 7 are fetched simultaneously and the cache is updated. Interleaving speeds up sequential accesses and increases the cache hit rate.

Procedure Sharing

PRIMOS IV and V users can write procedures (or subroutines) that can be shared by other users. They can also use shareable procedures or data bases that are part of PRIMOS or have been written by another user. The Prime 400 and 500 hardware, together with PRIMOS servicing "fault" conditions, preserves the integrity of the shared resource and controls the access rights granted by its originator.

When a procedure is shared, it exists only once on disk and, when active, only once in main memory regardless of the number of processes using it. For example, if several users are concurrently writing source code with the Text Editor program, and one user references a part (or a page) of the Editor that has already been brought into memory for some other user, that user is automatically linked to that same copy rather than transferring another "private" copy from disk into main memory.

For commonly used procedures, sharing greatly improves memory utilization by reducing the number of disk transfers required to bring in the program or data referenced by a process but not found in main memory (missing page faults). Sharing procedures means PRIMOS is more responsive to its users.

Re-entrancy. For a procedure to be shared by concurrent processes (re-entrant) and to be invoked by itself (recursive), its "pure" parts (those that do not change during execution) must be separated from those parts that do. This separation has already been done for PRIMOS IV and V's shareable resources and is

done automatically by the COBOL and FORTRAN translators when requested to do so by the user. Also, the Prime Macro Assembler (PMA) makes it easy for the assembly language programmer to write and use shared procedures.

Procedure Call. References to a shared procedure are made either by a user command (i.e., EDIT), by a language CALL PATCH statement in COBOL or FORTRAN, or a PCL NAME instruction in Assembly language. The language translators automatically generate code and variable areas that invoke and are compatible with the Prime 400 and 500's Procedure Call (PCL) mechanism. PCL is a hardware function that does all the housekeeping required to transfer control from one procedure (or subroutine) to another.

PCL saves and restores the state of the calling procedure and initializes the state of the called procedure. It creates and maintains 'stack' areas for data and argument variables and linkage areas for pointers. The PCL hardware performs these functions many times faster than would software equivalents. In addition, PCL guarantees that the access rights are followed, a function that cannot be performed by software. Thus, PCL and the concept of share procedures makes the Prime 400/PRIMOS IV and the Prime 500/PRIMOS V responsive and secure computing utilities.

Security

A combination of hardware and software creates a secure time-shared computer utility. PRIMOS IV and V protect the utility, its files, and its shared procedures from unauthorized access. The Prime 400 and 500 protect PRIMOS and users against unwanted intrusions or alterations by other users.

Log-In Procedure Protects Against Unauthorized Users. A user types "LOGIN UFDNAME" to gain access to the system. This will activate a LOGIN program (if one has been written) which cannot be defeated. The LOGIN program can add whatever security locks are deemed necessary and can activate accounting clocks that accumulate connect time, processor time, and disk transfer time until the user logs out (LOGOUT).

Passwords Protect User Files. The originator of a UFD (User File Directory) can define two passwords that must be satisfied when a user (or a process) tries to gain access to a file listed in the directory (ATTACH UFDNAME PASSWORD). One password is used by the owner, the other by a non-owner. For each file listed in the UFD, access rights can be defined by the owner; one set for the owner and one set for the non-owner. The rights granted depend on which password was used and they control read (and execute), write, delete, and truncate accesses.

Segment Descriptor Words (SDW) Protect Shared Procedures. When a procedure is made ready for execution (loaded), links are established to bind that procedure to all other procedures referenced. To do this the user must be able to ATTACH to the UFD that lists the referenced procedures (as file names) and be granted read access rights to the named files. Although such rights may be denied one user by another, all users can gain read access to the system library (UFD LIB) and the command library (UFD CMDNC0).

When the link is first established, the procedure is assigned to a segment by PRIMOS. All subsequent references to the procedure are linked to the same segment. The segment is defined by a Segment Descriptor Word (SDW) listed in a Descriptor Table.

The SDW contains the access rights granted by the procedure's owner to other users: read, write, execute, and gate. PRIMOS creates the SDW and, because of its unique privilege level, is the only process that can modify or delete an SDW. These rights are automatically transferred as a code to the Segment Table Look-aside Buffer (STLB), where they are used to control every memory reference.

Central Processor Ring Structure Protects PRIMOS From User. The SDW defines access rights for each of three rings or levels of privilege: Ring 0 is the most privileged which, by design, has all access rights and the right to execute all instructions. Ring 3 is the least privileged and does not have the right to execute those instructions that can alter the system's mode of operation (such as HALT). Ring 1 has access privileges between those of Ring 0 and 3, and the same instruction privileges as Ring 3. PRIMOS IV and V enjoy Ring 0 privileges; time-shared user processes, Ring 3. The ring number is a part of each calculated memory address.

The security provided by a ring structure can best be explained when one considers that an address of an instruction to be fetched may be defined for a different ring than the address of the data (or arguments) referenced by the instruction. For example, if a user's process (Ring 3) calls a shared procedure that is part of PRIMOS (Ring 0), the procedure's instructions are associated with Ring 0 and the data, or arguments (in stacks and linkage areas), with Ring 3. When such a shared procedure is called (using Procedure Call), reference is made to its Entry Control Block (ECB) that contains information required to initialize the procedure and start it running. The ECB is located in a segment whose SDW defines the access rights for a Ring 3 caller. If gate rights have been granted, the Procedure Call mechanism will strengthen the active ring number from 3 to 0 and allow the Ring 0 procedure to execute.

Otherwise, a fault condition occurs and the user's process cannot execute the procedure. This would happen for those procedures defined by PRIMOS to be for its own use and not to be shared by a user's process.

Weakening is an important function of the Procedure Call mechanism. The called and calling ring numbers are "OR"ed (3 + 0 = 3) and the resulting, weakened ring number is inserted into each argument (or pointer) transferred by the Procedure Call to the called procedure. When the called procedure is running and makes a reference to memory location pointed to by an argument, it is granted only the weakened Ring 3 privileges defined by the referenced SDW.

If a Ring 3 process references a location in the common address space, it would be granted the Ring 3 privileges previously defined by the 'owner' of the segment (usually PRIMOS) and would be denied those rights deemed by the owner to be unwise or harmful. Similarly, if the called Ring 0 procedure references a location in the private address space of the user's process, it would be granted only those privileges defined by the user for Ring 3.

The Prime 400 and 500's segmentation and Ring structure described above is the hardware that permits PRIMOS IV and V to offer a shared computing utility with complete and automatic access protection. The user need only define access rights and passwords, and then use passwords as required.

HOW PRIMOS IV AND V ARE ORDERED

PRIMOS IV and V are priced software products. Each may be ordered separately to run on suitable Prime 400 or 500 configurations or as a part of a packaged system designed for computational timesharing or interactive data processing. Several of the software subsystems supported by PRIMOS IV and V and described in this bulletin are priced separately and are also supported by PRIMOS III on Prime 300's with at least 128KB of main memory. Products that must be ordered individually are the COBOL translator, the RPG translator, each of the RJE emulators, and the PRIMENET network package. For detailed ordering information and availability schedules, please contact your local Prime Sales Representative or Prime's headquarters. Specifications may change as design improvements are introduced.

PRIME

PRIME Computer, Inc. 145 Pennsylvania Ave., Framingham, Mass. 01701