This paper is an overview of IBM's Virtual Machine Facility/370. It describes the virtual machine concept and its capabilities and implementation in VM/370. Two components of VM/370 are discussed—the control program and the Conversational Monitor System. The usefulness of VM/370 in multiple and diverse environments is covered. New developments in VM/370 from hardware assists to system extensions, networking, and handshaking are briefly described as an introduction to the rest of the papers in this issue.

VM/370—a study of multiplicity and usefulness

by L. H. Seawright and R. A. MacKinnon

The productivity of data processing professionals and other professionals can be enhanced through the use of interactive and time-sharing systems. Similarly, system programmers can benefit from the use of system testing tools. A systems solution to both areas can be the virtual machine concept, which provides multiple software replicas of real computing systems on one real processor. Each virtual machine has a full complement of input/output devices and provides functions similar to those of a real machine. One system that implements virtual machines is IBM's Virtual Machine Facility/370 (VM/370).[1]

VM/370 is an operating system that gives multiple users access to a computer by means of keyboard and display terminals for time sharing, system testing, production, and conversion. VM/370 manages the resources of a computer so that every user, local or remote, appears to have a complete replica of a System 370 including input/output (I/O) devices. Each user of VM/370 can select a different operating system, if desired, because different operating systems can run concurrently in different virtual machines.

This paper describes the capabilities of VM/370. For a historical perspective on VM/370 and the virtual machine concept, see References 2, 3, 4, and 5. One of the objectives of this paper and those that follow in this issue is to show how VM/370 has provided an architectural base for production as well as experimentation for a wide variety of users and installations.

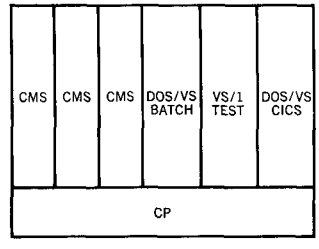
Virtual machine environments

VM/370 has two main components—the control program (CP) and the Conversational Monitor System (CMS). CP is the resource manager of the system. It creates virtual machines in which operating systems can run. It supports, as virtual machines, the operating systems that normally control real IBM System/360, System/370, and 303X (3031, 3032, 3033) processors. CMS provides an environment for interactive program development and personal computing, with functions such as language processors, editors, debugging tools, and applications packages.

control program

The operating systems supported in the VM/370 environment are versions of DOS/VS, OS/MFT, OS/MVT, DOS/VS, OS/VS1, SVS, MVS, and VM/370.[6] They execute under the control of CP, which manages the resources of the real system, giving each user or virtual machine access to appropriate I/O devices and the real central processing unit. CP multiprograms one virtual machine against another by using various time slicing algorithms and priorities, whereas batch systems would multiprogram tasks or partitions. Figure 1 illustrates multiple virtual machines that contain a variety of operating systems, including CMS, OS/VS1, and DOS/VS.

Figure 1 A VM/370 environment



While executing under CP, a virtual machine produces results that are functionally equivalent to those of a real machine, although execution is slower because the real machine is normally being shared. The role played by CP is transparent to a virtual machine, even though CP services are used continually to allow it to run. Interfaces between CP and the virtual machine are described in the Appendix.

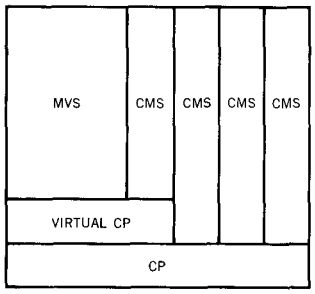
CP commands are available, if the user desires them, to enhance the control and debugging of the operating system in the virtual machine environment. These commands provide for displaying storage of the virtual machine (not real storage), setting instruction address stops, and dumping virtual storage. In essence, these are the functions a system programmer would perform at the console of a real CPU.

During virtual machine execution, all virtual machine code (both supervisor and problem state) executes unchanged from that which would run on a real machine. It relies on the normal operating system files and data sets to execute, and it communicates with the virtual machine operator through a console, usually a terminal. Thus the messages displayed on the console are identical to those that would be produced by a real machine.

A virtual machine assumes total responsibility for the management of jobs or function within it. Therefore, the virtual machine depends largely on its own operating system for access methods

and services such as error recovery, multiprogramming and multitasking control, spooling (other than that which CP can provide), demand paging (if the virtual machine is running a virtual system), job initiation and termination, and handling of abnormal situations. CP provides only those services required to resolve differences between the virtual machine and the real system, to dispatch virtual machines, and to handle the real system. Unless certain communications interfaces are utilized,[7] virtual machines execute with a high degree of isolation and protection because of the virtual machine architecture.[8]

Figure 2 VM/370 running in a virtual machine



Just as OS/VS and DOS/VS can execute in a virtual machine, CP can execute in a virtual machine to provide a virtual VM/370 environment. This environment is particularly useful in debugging parts of CP or introducing experimental or maintenance versions of CP into a production system. For example, at the IBM Cambridge Scientific Center, an experimental virtual machine was used to test proposed performance enhancements for MVS in the VM/370 System Extensions Program Product.[9] The real system was a System/370 Model 158 uniprocessor. A virtual machine on the Model 158 was brought up, and an experimental copy of CP was loaded into the virtual machine. A virtual MVS machine and CMS were then executed under virtual CP to test the proposed performance enhancements for MVS. During execution in the virtual machine, virtual CP dispatched virtual machines in the same manner as the real CP. All of this activity appeared as one virtual machine to real CP, as shown in Figure 2.

Conversational Monitor System (CMS)

CMS provides program development and personal computing functions in an interactive fashion to an individual terminal user. That is, CMS supports a single user of VM/370 at a single terminal, rather than supporting multiple terminals. Access to multiple terminals is accomplished by CP's ability to support multiple CMS virtual machines—one for each interactive user. Although each CMS user occupies a separate virtual machine, code is shared among CMS machines. CMS is designed specifically for VM/370 and depends on CP for its execution. Thus it cannot operate independently on a real machine, as can the other operating systems discussed above.

CMS provides both problem solvers and application programmers with the language processors and compilers normally associated with OS/VS and DOS/VS.[10] It also provides many of the file access methods associated with those operating systems. Consequently, CMS can be used for execution of applications that might otherwise run in another virtual machine under DOS/VS or OS/VS. CMS also has editing, text formatting, and debugging capabilities; it includes command procedures and application packages; and it provides for interactive execution of user programs.

Table 1 Type of access allowed by CMS, for specified access methods, to CMS, OS, and DOS disk files

Access method	*CMS format*	*OS format*	*DOS format*
VSAM	—	read/write	read/write
SAM	read/write	read only	read only
BDAM	read/write	—	—
PAM	read/write	read only	—

In addition to its use of many standard OS/VS and DOS/VS access methods, CMS supports its own file system, which has a format unlike that of OS or DOS. CMS assumes total responsibility for file management, including blocking and deblocking. The user of CMS deals with files only on a named basis. Individual file space is not pre-allocated, but is obtained and deleted dynamically. At any time, the user can query CMS as to the amount of remaining free space. CMS also gives the user access to OS and DOS disks, the formats of which are controlled by the user through the appropriate OS and DOS access methods. The types of access allowed for each type of disk are listed in Table 1.

CP provides disk space for CMS and other virtual machines by means of minidisks, which are predefined sets of contiguous cylinders on a disk. A minidisk normally is considered a subset of a full disk, even though it can be an entire disk. CP maps the user's input and output to the disk. Minidisks can be shared with other users on a VM/370 system or reserved exclusively for a particular virtual machine.

CMS provides for terminal-based, interactive execution of applications unless the user chooses to logically disconnect his terminal and run his application in the background. An APL interpreter, VS/BASIC, PL/I, and FORTRAN also provide a high degree of interactivity and problem solving capability for the CMS user.[11,12,13]

CMS interfaces to the terminal user through a series of commands, rather than through a job control language as in other operating systems. Because of the file support provided by CMS, little operating system knowledge is required on the user's part other than familiarity with the activities directly related to accomplishing the desired function. Thus CMS can be said to present a "user-friendly" interface to the user. Job control is eliminated, punched cards are unnecessary, and terminal prompting provides for error correction at the source. There are no turn-around delays like those associated with batch processing and remote job entry. Doherty and Kelisky[14] describe the evolution of CMS interactive computing services at the IBM Thomas J. Watson Research Center and their advantages for users.

Usefulness of VM/370

VM/370 can be used for a wide variety of purposes. It is important to understand that there is no typical user of VM/370; to try to find such a user is to overlook the system's most valuable attribute—accommodation of diverse computing environments. Discussed below are some of the purposes for which VM/370 is used.

multiple operating systems

Because it can run multiple operating systems concurrently, VM/370 can be used to advantage when an installation is converting from one production operating system to another. The old system can continue to run while the new one executes with converted programs in a separate virtual machine. CMS can be used as an interactive tool to make the necessary program changes and test the modified code.

Similarly, maintenance changes can be applied to the production operating system or to application programs. As previously noted, this same approach can be applied to VM/370 itself. In effect, maintenance is viewed as conversion, and the use of virtual machines ensures that the production work load can be handled without interruption.

Occasionally, a special application program may have to be written to execute in some operating system other than the production system. With virtual machines, the application can be run without dedicating the real system to the other operating system. This approach can be used when certain applications are to be phased out eventually and not converted to a new operating system.

Some installations run multiple copies of the same operating system to obtain additional function—for example, to increase the number of executable job partitions, to enhance performance, or to isolate operating systems and application programs from each other.

address space isolation

Every virtual machine has its own virtual address space, which is accessed through dynamic address translation hardware and which uses a unique set of segment and page-translation tables. These attributes can be important in minimizing the effects of software failure either in the operating system or in application code. It is this isolation of address spaces that helps VM/370 run multiple operating systems, as discussed above.

security

Where there is concern about access to sensitive data, or where there has been unwillingness to allow access via terminals, the virtual machine architecture of VM/370 provides a high degree of system integrity and security.[8] In addition to providing address space isolation, VM/370 requires a password when each user logs

on. Minidisks can be protected by passwords for writing, reading, or both. Further, VM/370 provides several levels of protection for the virtual machine description, which is protected in the system directory.

unattended operation

The real system controlled by VM/370 can operate essentially unattended, unless the mounting of tape or disk volumes is required, or a catastrophic error occurs. Few messages requiring operator intervention are presented to the system console, and the system provides for automatic restarting of CP. For instance, after a blizzard in February 1978, the VM/370 system at the Cambridge Scientific Center, which services many remote users, operated for a week with the computer room locked and the operator's console locked and logically disconnected. The system continued to provide service and was monitored remotely from a terminal with operator privileges.

program development

Because CMS has a full complement of language processors, compilers, file access methods, command procedures, debugging capabilities, and editors, a terminal user can create, document, compile, test, and debug programs in a true interactive environment. A dedicated real machine is not required, and system and application programmers alike can benefit from increased productivity. Program development can be combined with time sharing and production work under central management and operation.

problem solving for end users

Under CMS, interactive tools such as VS/BASIC and APL are available for commercial time sharing and management science applications, or, at a university, for student time sharing and academic computing. End user applications can range from computer-assisted instruction to departmental reporting systems and query facilities, from word processing to electronic mail. Many enterprises have used VM/370 for time sharing to make use of excess capacity or to achieve economy of scale by expanding the installed configuration.

production batch processing

Many installations run a batch processing system under VM/370, along with other uses of virtual machines. CMS also has a batch capability which can be used for production work. The final section of this paper, on virtual machine performance, discusses the options by which CP can give higher priority to batch machines when maximal throughput is desired.

computer science

VM/370 can provide a computer science laboratory on a single machine. Its isolation and ease of use, and the great variety of available system and application programs, make it a capable host for computer science applications. At the Cambridge Scientific Center, for instance, VM/370 provides for interactive computing and experimentation by the staff and for remote time sharing by IBM

subscribers throughout the United States, and it is a part of a networking facility that services many IBM locations.[15] Remote time sharing and networking are considered production operations in that high levels of service and function are provided to the user. With this type of function, VM/370 can significantly extend lines of communication among technologists who may be hundreds of miles apart.

New developments

Changes have been made in specific VM/370 implementations which have affected the function and performance of the total system. This discussion thus far has stressed the isolation of virtual machines. Subsequent papers, by MacKinnon on architectural changes,[7] Jensen on inter-virtual-machine communication,[16] Hendricks and Hartmann on the networking capability,[17] and Attanasio on Virtual Control Storage,[18] relate to the increasing trend toward communication among virtual machines and the advent of subsystem architectures that exploit the virtual machine environment.

The manner in which VM/370 supports the problem solving language APL is a significant development. VS/APL under CMS uses the APL microcode assist, which is available on certain models of System/370.[7] Also recently introduced is attached processor support in VM/370 for Models 158 and 168 and the 3031 processor, providing processing power beyond the capabilities of a single CPU. Such additional power might be required, for example, by a CMS user with CPU-bound applications that can benefit from additional instruction processing capabilities. Holley et al.[19] describe the architecture and implementation of multiprocessing within an environment originally designed only for uniprocessing.

Total system performance has always been a concern when considering use of VM/370. The overhead imposed by the hypervision and simulation activities of CP for non-CMS virtual machines normally decreases batch throughput and increases the response time of these machines compared with the performance of a real machine. New developments have occurred in this area and are discussed below, along with the ways in which VM/370 normally addresses performance.

performance options

A single virtual machine can run in virtual-equals-real mode, whereby the virtual machine's real storage is not demand-paged by CP. Specified page frames can be locked into real storage. Devices and channels can be allocated to a virtual machine on a dedicated basis. And executable code can be shared among virtual machines.

For scheduling of virtual machines, CP employs either a biased scheduler[20] or a resource manager[9] that uses the fair-share scheduling concept, by which CP can give preferential service to a particular virtual machine to enhance its throughput. CP executes in its own address space without dynamic address translation (but in extended control mode). CMS also operates without dynamic address translation, but in basic control mode. Thus certain levels of paging overhead incurred by other virtual system control programs are not incurred by CP or CMS. These performance options generally are available through software on all processors capable of running VM/370.

VM/370 has a system performance measurement facility, which provides a method of obtaining system resource utilization on line while CP is running, as well as collecting measurement data for later analysis. A methodology for analyzing system performance is described by Tetzlaff.[21]

handshaking

Other software developments are more specialized and involve changes to VM/370 and other system control programs, such as DOS/VS and OS/VS1, that utilize the virtual machine environment. These changes allow the operating system in a virtual machine to recognize that it is running under VM/370 and to communicate with CP. Thus OS/VS1 and DOS/VS virtual machines have a direct interface to CP, so they no longer have to perform operations that are redundant when they are operating in a virtual machine environment. The results of these changes, collectively termed *handshaking*, are greater operational efficiency and improved virtual machine performance. Handshaking is discussed in some detail by MacKinnon.[7]

hardware assists

Certain models of the System/370 and 303X processors incorporate hardware designed to handle most-frequently-executed CP functions in order to reduce the overhead associated with CP hypervision and to enhance virtual machine performance. This assistance, provided by virtual machine assist and Extended Control Program Support (ECPS) hardware, also is discussed by MacKinnon.[7]

Summary

This introductory paper has attempted to explain the essential elements of VM/370 structure and the interfaces it provides for virtual machines. Throughout, an effort has been made to show why VM/370 is used, how it encompasses a multiplicity of uses, and where development has progressed over the years.

The things that have changed most about VM/370 are how it is viewed and the uses to which it has been put within many dif-

ferent installations. What has changed least is its ability to be installed rapidly and become productive in many circumstances and for many purposes.

ACKNOWLEDGMENTS

The authors express their appreciation to the many people who, over the years, have helped with an understanding of the uses, development, potential, and future of virtual machines. Too numerous to name individually, they represent many activities within IBM and many non-IBM organizations such as GUIDE and SHARE, and they include many individual users. They continue to enhance our understanding of how the virtual machine concept helps them achieve their data processing objectives.

Appendix: CP interfaces to virtual machines

The architecture of the virtual machines that run under CP is defined to be that of IBM System/370. Thus there is almost total commonality between the instruction sets of the virtual machine and the real machine that executes real CP. While VM/370 runs on the 303X processors, CP itself does not use the 14 new instructions that extend the System/370 instruction set to those processors. However, if the MVS System Extensions Program Product[22] is running as a virtual machine on any of the 303X processors, MVS can execute the new instructions.[7] If any non-System/370 emulation is run in the virtual machine, it is controlled by an operating system emulator program, which appears as a normal System/370 program to CP.

CP code executes on the real machine in the real supervisor state. When a virtual machine is dispatched, or run by CP, it executes on the real machine in the real problem state. For each virtual machine, CP maintains a central control block, called the VMBLOK, in its nucleus. Among the contents of the VMBLOK are a virtual program status word, virtual general-purpose and floating-point registers, and information on whether the virtual machine is operating in the virtual supervisor or virtual problem state. Thus the virtual machine operating system continues to alternate between supervisor and problem state as it would when in control of a real machine. Only CP recognizes that, in fact, the virtual machine executes in the real problem state at all times. The virtual operating system can issue no instruction (or sequence of instructions) that will reveal that CP is, in fact, running the virtual machine in the real problem state, except in the case of handshaking.[7]

CP control

A virtual machine executes System/370 instructions directly on the real machine unless CP gains control because of an interruption by an asynchronous event or by a supervisor call or program exception. Examples of asynchronous events are I/O interruptions and timer interruptions associated with the completion

of a time slice. Program exception interruptions occur whenever a virtual machine tries to execute a privileged instruction. The most likely source of privileged instructions is within the operating system code of the virtual machine, but CP can also handle privileged instructions in application code. Thus until CP completes its analysis, it does not regard program exception interruptions as unusual. Program interruptions are the chief interfaces between the virtual machine and CP that allow CP to gain control when virtual operating system services have been requested.

DIAGNOSE is a privileged instruction used by some virtual machines as a specially-defined interface to CP. In a sense, DIAGNOSE is a special-purpose supervisor call that allows virtual machines to request CP services. CMS, being dependent on CP, is the most common user of the DIAGNOSE interface, but modifications have been made to DOS/VS and OS/VS1 to allow them to signal CP with DIAGNOSE. This use of DIAGNOSE is covered by MacKinnon.[7]

Once CP gains control, an analysis is performed to determine the reason for the program exception generated by the virtual machine. If the virtual machine was operating in the virtual problem state, CP passes or reflects the real program interruption to the virtual machine, which thus regains control and proceeds through its analysis and handling of the interruption. If the virtual machine was in the virtual supervisor state, CP simulates execution of the privileged instruction in the following manner: First, it determines what function the virtual machine is trying to perform. If input or output is involved (START I/O, HALT I/O, or TEST I/O), CP must make any necessary adjustments in mapping between the virtual and real devices. CP translates virtual machine channel command words (CCW's) into real CCW's to reflect the real storage page frames to be used. It uses the indirect data addressing facility of System/370 in doing so. In the case of I/O from or to minidisks, CP must map logical track addresses to physical (real) track addresses to get to the proper minidisk space. CP then schedules the I/O operation. Control ultimately returns to the virtual machine's operating system, which proceeds as if the I/O had been initiated.

CP attempts to return control to the virtual machine as soon as it has completed its simulation process. The objective is to allow the virtual machine to complete its time slice with minimal interruption. When the simulation cannot be completed (as when I/O cannot be scheduled immediately), CP dispatches another virtual machine in its run list.

Other interruptions most commonly involve some process or operation for which CP is responsible. When they involve an activity

undertaken on behalf of a virtual machine (such as I/O), CP turns control over to the virtual machine's operating system, just as the real hardware would indicate completion of an activity. Thus in the case of I/O, the virtual machine (when enabled for this interruption) has its old program status word and a correct channel status word stored, and control passes to its I/O interruption handler.

enhanced operation

Finally, CP undertakes a variety of operations as part of the services it provides to enhance virtual machine operation. These operations are described below:

demand paging

CP performs demand paging of its real storage outside the fixed nucleus. Consequently, virtual machines such as DOS/VS, OS/VS1, SVS, and MVS, which employ demand paging for their virtual address space, are subject to two levels of paging: that initiated by the virtual machine (and regarded by CP as normal I/O) and that undertaken by CP (which is transparent to the virtual machine). Therefore, operating systems such as OS/MVT, which do not employ demand paging, are paged by CP. Performance options provided by CP, however, permit a particular virtual machine to run in virtual-equals-real mode if no paging by CP is required for the virtual machine. CP's performance options also permit individual page frames to be locked and some virtual machines to dispense with redundant paging through handshaking.

spooling

CP provides a spooling system that handles unit-record I/O from or to the virtual machine. This system does not negate the use of any spooling system run in the virtual machine as part of the virtual operating system. In fact the virtual spooling system can be used exclusively by the virtual machine in place of CP spooling. CMS relies on CP spooling, however, since no such facility is integrated into its code.

DASD management

CP provides for DASD (direct access storage device) management in a variety of ways:

- A DASD volume can be dedicated to a particular virtual machine and be under the complete control of that machine. No sharing is implied.
- CP can divide a DASD volume into subsets called minidisks, each of which appears to the virtual machine operating system as a complete physical device but normally contains fewer cylinders. CP maps minidisks to real disks, but it is up to the virtual machine operating system to manage the space within the bounds of the minidisks. (CP does not allow the virtual machine to get outside the bounds.) Minidisks can be shared among machines on a read-only basis. With proper controls, they can also be shared for writing.

- CP can share a DASD volume among several virtual machines by means of the minidisk concept, as well as by supporting RESERVE/RELEASE for OS virtual machines.

console management

CP maintains responsibility for the terminal used by each virtual machine as an operator's console. The terminal may be locally attached, on a communications link, or disconnected by the user when a console is not needed.

real hardware control

CP maintains complete control of the real hardware system and manages real storage. It initiates I/O, handles first-level interruptions, and recovers from errors caused by machine checks. CP recovers from errors in I/O that it initiates, including paging, spooling, and DIAGNOSE I/O, but it reflects other I/O errors back to the appropriate virtual machine. CP may terminate the affected virtual machine rather than terminate the entire real machine. CP manages the real storage in 4K-byte page frames and 64K-byte segments but allows each virtual operating system to manage its virtual storage in its own way. (DOS/VS and OS/VS1, for example, use virtual page frames of 2K bytes, and DOS/360, OS/MFT, and OS/MVT do not make use of demand paging.)

CITED REFERENCES AND NOTES

1. *IBM Virtual Machine Facility/370 Introduction*, IBM Systems Library, order number GC20-1800, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.
2. R. A. Meyer and L. H. Seawright, "A virtual machine time-sharing system," *IBM Systems Journal* **9**, No. 3, 199–218 (1970).
3. R. P. Parmelee, T. I. Peterson, C. C. Tillman, and D. J. Hatfield, "Virtual storage and virtual machine concepts," *IBM Systems Journal* **11**, No. 2, 99–130 (1972).
4. M. A. Auslander and J. F. Jaffe, "Functional structure of IBM virtual storage operating systems—Part I: Influences of dynamic address translation on operating system technology," *IBM Systems Journal* **12**, No. 4, 368–381 (1973).
5. R. P. Goldberg, "Survey of virtual machine research," *Computer*, June 1974, pages 34–35.
6. *IBM Virtual Machine Facility/370: Operating Systems in a Virtual Machine*, IBM Systems Library, order number GC20-1821, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.
7. R. A. MacKinnon, "The changing virtual machine environment—Interfaces to real hardware, virtual hardware, and other virtual machines," *IBM Systems Journal* **18**, No. 1, 18–46 (1979, this issue).
8. J. J. Donovan and S. E. Madnick, "Virtual machine advantages in security, integrity, and decision support systems," *IBM Systems Journal* **15**, No. 3, 270–278 (1976).
9. *IBM Virtual Machine Facility/370 System Extensions General Information Manual*, IBM Systems Library, order number GC20-1827, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.
10. CMS language processors are provided as follows:
    Assembler: OS/VS, DOS/VS, VM/370 Assembler.
    COBOL: OS/VS COBOL, OS ANS COBOL Version 4, OS COBOL Interactive Debug, DOS/VS COBOL.
    FORTRAN: OS Code-and-Go, OS FORTRAN IV (G1), OS FORTRAN IV (H) Extended, FORTRAN H Extended Optimization Enhancement, FORTRAN Interactive Debug.

PL/I: OS PL/I Optimizing Compiler, OS PL/I Checkout Compiler, DOS PL/I Optimizing Compiler.
VS/BASIC: VS/BASIC Processor.
VS/APL: VS/APL Interpreter.

11. *VS APL: General Information*, IBM Systems Library, order number GH20-9064, IBM Corporation, P.O. Box 50020, Programming Publishing, San Jose, California 95150.
12. *VS BASIC General Information*, IBM Systems Library, order number GC28-8302, IBM Corporation, P.O. Box 50020, Programming Publishing, Palo Alto, California 95150.
13. *IBM FORTRAN Program Products for OS and the CMS Component of VM/370 General Information*, IBM Systems Library, order number GC28-6884, IBM Corporation, Programming Publishing, 1271 Avenue of the Americas, New York, New York 10020.
14. W. J. Doherty and R. P. Kelisky, "Managing VM/CMS systems for user effectiveness," *IBM Systems Journal* **18,** No. 1, 143–163 (1979, this issue).
15. R. P. Crabtree, "Job Networking," *IBM Systems Journal* **17,** No. 3, 206–220 (1978).
16. R. M. Jensen, "A formal approach for communications between logically isolated virtual machines," *IBM Systems Journal* **18,** No. 1, 71–92 (1979, this issue).
17. E. C. Hendricks and T. C. Hartmann, "Evolution of a virtual machine subsystem," *IBM Systems Journal* **18,** No. 1, 111–142 (1979, this issue).
18. C. R. Attanasio, "Virtual Control Storage—security measures in System/370," *IBM Systems Journal* **18,** No. 1, 93–110 (1979, this issue).
19. L. H. Holley, R. P. Parmelee, C. A. Salisbury, and D. N. Saul, "VM/370 asymmetric multiprocessing," *IBM Systems Journal* **18,** No. 1, 47–70 (1979, this issue).
20. *Virtual Machine Facility/370 Features Supplement*, IBM Systems Library, order number GC20-1757, IBM Corporation, Department 824, 1133 Westchester Avenue, White Plains, New York 10604.
21. W. H. Tetzlaff, "State sampling of interactive VM/370 users," *IBM Systems Journal* **18,** No. 1, 164–180 (1979, this issue).
22. *OS/VS2 MVS/System Extensions General Information Manual*, IBM Systems Library, order number GC28-0872, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.

GENERAL REFERENCES

In addition to the cited references, the publications listed below address many details touched on in this paper and may enhance the reader's understanding of VM/370 and its facilities.

IBM Virtual Machine Facility/370: Planning and System Generation Guide, IBM Systems Library, order number GC20-1801, IBM Corporation, P.O. Box 390, Poughkeepsie, New York 12602.

IBM Virtual Machine Facility/370: CMS User's Guide, IBM Systems Library, order number GC20-1819, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.

IBM Virtual Machine Facility/370 Basic System Extensions General Information Manual, IBM Systems Library, order number GC20-1828, IBM Corporation, Department D58, P.O. Box 390, Poughkeepsie, New York 12602.

Network Job Interface General Information Manual, IBM Systems Library, order number GH20-1941, IBM Corporation, Department 825, 1133 Westchester Avenue, White Plains, New York 10604.

VM/370 Networking Program Reference and Operations Manual, IBM Systems Library, order number SH20-1977, IBM Corporation, Department 825, 1133 Westchester Avenue, White Plains, New York 10604.

Y. Bard, "An analytic model of the VM/370 system," *IBM Journal of Research and Development* **22,** No. 5, 498–508 (September 1978).

P. H. Callaway, "Performance measurement tools for VM/370," *IBM Systems Journal* **14,** No. 2, 134–160 (1975).

J. J. Donovan, *Use of Virtual Machines in Information Systems*, Report No. MIT-EL-75-010, MIT Energy Laboratory, Cambridge, Massachusetts 02139 (May 1975).

J. J. Donovan and S. E. Madnick, "Hierarchical approach to computer system integrity," *IBM Systems Journal* **14,** No. 2, 188–202 (1975).

C. Y. Lam and S. E. Madnick, *Use of Virtual Machines for Development of Decision Support Systems: Strategies for Interfacing Virtual Machines, Internal Report No. R001-7804-01*, Center for Information Systems Research, MIT Sloan School of Management, 50 Memorial Drive, Cambridge, Massachusetts 02139 (March 1978).

A. J. Smith, "Bibliography on paging and related topics," *ACM Operating Systems Review* **12**, No. 4, 39–56 (October 1978).

ACM SIGARCH-SIGOPS Workshop on Virtual Computer Systems (Harvard University, March 1973), ACM, 1133 Avenue of the Americas, New York, New York 10036.

Reprint Order No. G321-5084.