A. Padegs

System/370 Extended Architecture: Design Considerations

This paper reviews the overall objectives and the design considerations that led to the System/370 Extended Architecture (370-XA). It presents an overview of the differences between the System/370 and 370-XA architectures and summarizes all architectural extensions, deletions, and changes. Then it describes in more detail the extensions for interpretive execution, 31-bit addressing, protection, tracing, inter-CPU communications, and extended-precision floating-point division.

Introduction

The System/370 Extended Architecture (370-XA) is a major step in the evolution of the System/360 architecture originally introduced in 1964 and expanded into System/370 architecture in 1970. The development of the 370-XA architecture started in 1975. At that time, in recognition that the system-control facilities of the machine must match the requirements of the operating system, a further extension of the System/370 architecture [1] was undertaken in two directions.

First, an architecture for small-to-medium-size machines was designed that met the requirements of the Disk Operating System (DOS). This was called the architecture for ECPS:VSE mode (Extended Control Program Support: Virtual Storage Extended) [2], and it was first made available in 1979 on the 4300 processors. The main change from System/370 was the substitution of the one-level-addressing facility for the System/370 dynamic address translation. The DOS/VSE operating system offers one virtual address space of 16M bytes [$M = 2^{20}$], and the architecture is simplified accordingly by not providing the multiple-address-space capability of the System/370.

Second, an architecture for medium-to-high-end machines, called 370-XA [3], was designed that accommodated the Multiple Virtual Storage (MVS) operating system. Here the emphasis was on the following: communication between multiple virtual-address spaces, larger virtual-address spaces, larger real storage, and the interconnection of CPUs within one system controlled by a single copy of an operating system. Within the past few years, some improvements toward these objectives were also included in the System/370 architecture, but the structural changes offering 31-bit addressing are available only in 370-XA. The emulation facilities of 370-XA were specifically developed to permit efficient implementation of virtual machines, based on experience gained with the Virtual Machine/370 (VM/370) operating system.

Most 370-XA facilities were made generally available earlier this year (as of the date of this publication) on the 3081 and 3083 machines. Facilities for emulation, including the instruction START INTERPRETIVE EXECUTION, have also been announced and will be made available later.

In the following sections, first the specific objectives of 370-XA are listed, followed by a description of the constraints on the development and a few remarks on how the new architecture is invoked. Then, an overview is presented of the differences between System/370 and 370-XA in terms of 1) the new facilities which have been added to those available on System/370, and 2) the obsolete or redundant facilities deleted. Subsequently, the major architectural extensions are described, and the motivation for their design is reviewed. The design considerations of the channel subsystem are covered in the accompanying paper by Cormier, Dugan, and Guyette [4].

Objectives and constraints

The key requirement for the 370-XA architecture was to extend the storage-addressing capability of System/370. By the mid-1970s, System/370 Model 168 multiprocessing systems had already implemented the maximum addressable storage of 16M bytes, and the architecture needed to be extended to allow attachment of additional physical storage.

The problem was partially eliminated with the introduction of the extended-real-addressing facility, first shipped in October 1981 on the 3033 and 3081 processors. This facility made use of two unused bit positions in the page-table entry to extend to 26 bits the real storage address provided by the dynamic-address-translation (DAT) mechanism, thus offering addressability up to 64M bytes of main storage. This extension was available, of course, only to addresses subject to DAT. It did not permit placing DAT and other control tables above the 16M-byte boundary, did not apply to the base and index components from which addresses are generated, and did not extend the addressability of virtual storage.

Another requirement was to extend the size of the virtual storage. Even with the introduction of the dual-address-space facility on the System/370 and the associated cross-memory services offered by MVS, which allowed software subsystems to be placed in their own virtual-address spaces, growth in the virtual-storage requirements for both the operating system and applications required the extension of the virtual-address space beyond 16M bytes.

A major overall requirement was to improve the efficiency, performance, and reliability of large systems on the basis of our experience, considering the available technology and anticipating new applications. Specifically,

1. Increases in the projected customer requirements for processing power could not be met solely through advances in technology and CPU design; the required processing power could be offered only by interconnecting into one system a number of processors operating under a single operating system. Considerations that would permit efficient operation of a number of CPUs sharing main storage were major influences in the development of the 370-XA architecture for channels and tracing.
2. Further efficiencies in the use of multiple processors could be obtained by transferring a larger part of the I/O and system-control functions to special-purpose processors, such as channel and service processors, thus freeing the general-purpose CPU for executing application programs. Also, the transfer of additional functions to the channel processor would reduce the number of I/O interruptions and instructions (with their associated overhead), and the execution of some auxiliary functions by the service processor would make it possible to delete some functions in the CPU machine-check architecture.
3. Recognition of the value of the virtual-machine concept and the observation that the VM/370 operating system had attained a wide usage led to the development of the 370-XA interpretive-execution architecture as a replacement for the various virtual-machine assists offered on System/370.

The 370-XA architecture was developed subject to the following constraints:

1. Within the limitations described in the *IBM System/370 Principles of Operation* [5], machines designed to the 370-XA architecture had to provide upward compatibility for application programs written to run according to the System/370 architecture.
2. The extent of change in the facilities pertaining to the control program was limited by the number and cost of the associated changes acceptable in the operating system.
3. It had to be possible to attach and operate most types of System/370 I/O devices with 370-XA.

Control

In contrast to the program-controlled mode bit introduced in System/370 to allow selection between the extended-control (EC) and basic-control (BC) modes of operation [5], 370-XA is considered a new architecture, and the choice between System/370 and 370-XA is made manually. This approach was necessary because of the large number of facilities changed and replaced, particularly those associated with input and output. It simplified both the new architecture and the machine design by permitting certain superseded facilities to be eliminated from the new architecture, and by eliminating the otherwise necessary specification of the interaction between various combinations of old and new architectural facilities. Manual selection was acceptable because use of the affected facilities normally is limited to a central part of the control program, which runs with either the old or new facilities but does not alternate between them.

By having both the System/370 and 370-XA modes available in the machine by manual selection, the transition to the 370-XA mode of operation can be made under the control of installation management. Once the system has been set up and debugged to operate with a control program that uses 370-XA mode and offers virtual-machine facilities, control programs requiring the System/370 mode of operation can be executed in interpretive-execution mode under the control of the 370-XA control program and without manual intervention.

Summary of differences with System/370

In describing an architecture, it is convenient to consider the entire architecture as being made up of increments called

Table 1 New 370-XA facilities.

New facility	*Number of instructions*	
	Unprivileged	*Privileged*
Channel subsystem	—	13
Interpretive execution	—	1
31-bit addressing (logical and real)	3	—
Page protection	—	—
Tracing	—	1
SIGNAL PROCESSOR extensions	—	—
DIVIDE (extended precision)	1	—
Totals	4	15

facilities. Table 1 lists the major new architectural facilities introduced with 370-XA and the number of instructions associated with each. Table 2 lists the new instructions and their key attributes.

Computer architectures have a tendency to grow because, when new facilities are added, old ones are preserved for the sake of compatibility, thus burdening the system with obsolete or redundant instructions and formats. The 370-XA architecture counters this trend by not providing a number of facilities that were available in System/370 but that have been superseded by improved or functionally richer versions. All of these eliminated facilities pertain to the control program, and their absence should not affect compatibility for application programs.

The deleted facilities are listed in Table 3. The 370-XA channel subsystem supersedes the System/370 channels and channel-set switching. Segment protection has been superseded by page protection, providing protection with finer block size against unauthorized storing. The basic-control mode and the interval timer were supplemented by improved facilities in System/370, but both the old and the new were offered concurrently on System/370 machines to ease the changeover. Some System/370 facilities are not provided because they have been found to be not needed for the current customer applications and system configurations (for example, direct control and external signals) or for the new control-program structure (for example, the 2K-byte [K = 2^{10}] page and protection-block size and the 64K-byte segment size). The *program-reset* and *initial-program-reset* orders of the SIGNAL PROCESSOR instruction caused the channels to be reset and thus were not applicable to the 370-XA architecture with a channel subsystem shared by CPUs. The *initial-microprogram-load* order was intended primarily for restoring microcode after failure but proved not to be needed in an environment that depends on the service processor for maintenance and recovery.

Additionally, six System/370 machine-check indicators and fields are not provided. They pertain either to a deleted facility (for example, interval timer) or to procedures where the new generation of machines performs recovery without bringing the condition immediately to the attention of soft-

Table 2 New instructions.

Instruction name	*Mnemonic*	*Type*		*Op code*
General instructions				
BRANCH AND SAVE AND SET MODE	BASSM	RR	Unpriv.	0C
BRANCH AND SET MODE	BSM	RR	Unpriv.	0B
DIVIDE (extended precision)	DXR	RRE	Unpriv.	B22D
INSERT PROGRAM MASK	IPM	RRE	Unpriv.	B222
Control instructions				
TRACE	TRACE	RS	Priv.	99
START INTERPRETIVE EXECUTION	SIE	S	Priv.	B214
I/O instructions				
CLEAR SUBCHANNEL	CSCH	S	Priv.	B230
HALT SUBCHANNEL	HSCH	S	Priv.	B231
MODIFY SUBCHANNEL	MSCH	S	Priv.	B232
RESET CHANNEL PATH	RCHP	S	Priv.	B23B
RESUME SUBCHANNEL	RSCH	S	Priv.	B238
SET ADDRESS LIMIT	SAL	S	Priv.	B237
SET CHANNEL MONITOR	SCHM	S	Priv.	B23C
START SUBCHANNEL	SSCH	S	Priv.	B233
STORE CHANNEL PATH STATUS	STCPS	S	Priv.	B23A
STORE CHANNEL REPORT WORD	STCRW	S	Priv.	B239
STORE SUBCHANNEL	STSCH	S	Priv.	B234
TEST PENDING INTERRUPTION	TPI	S	Priv.	B236
TEST SUBCHANNEL	TSCH	S	Priv.	B235

ware. Also, a number of control-register positions and storage assignments associated with the discontinued facilities are no longer required.

To ease the adaptation of System/370 control programs for operation with 370-XA, the facilities that were carried forward from System/370 to 370-XA were left as much as possible unchanged. In a few cases, however, the definitions were altered. All of these changes pertain to control-program functions and are identified in the *IBM 370-XA Principles of Operation* [3]. They include such aspects as the effect of prefixing on LOAD REAL ADDRESS when translation tables are in the storage area subject to prefixing.

The October 1981 edition of the *IBM System/370 Principles of Operation* contained a total of 208 instructions. After subtraction of the 17 deleted instructions, as well as MOVE INVERSE [6], and addition of the 18 new instructions (excluding START INTERPRETIVE EXECUTION), the initial *IBM 370-XA Principles of Operation* also contains 208 instructions.

Interpretive execution

Next to the channel subsystem, the facility for interpretive execution forms the largest extension incorporated into 370-XA. This facility permits establishing a virtual machine for execution of either System/370 or 370-XA programs.

A virtual machine is established and its operation is initiated by the execution of the instruction START INTERPRETIVE EXECUTION (SIE). This instruction designates a control block in main storage, called the state description, which specifies the state and the architecture of the virtual machine on which a program, called the guest program, is to be executed (the program issuing SIE is referred to as the host program). The state description includes such guest information as the program-status word, prefix, and timing-facility values and specifies what part of the host's virtual address space the guest occupies.

Most guest functions are performed by the virtual machine. When a condition is encountered that requires intervention by the host program, interception occurs: The machine leaves the interpretive mode and transfers control to the host at the location following the SIE instruction. This includes such conditions as execution of some privileged instructions, notably the I/O instructions and DIAGNOSE, handling of certain guest interruptions, and setting up the wait state in the guest PSW. The machine leaves the interpretive mode also when an external or I/O condition causes the host to be interrupted.

Whenever the machine leaves the interpretive mode, the state description is updated to reflect the current state of the guest program so that, subsequently, execution of the guest can be resumed without further changes to the state description. When interception occurs, information is placed in the state description to facilitate identification and analysis of the guest condition by the host program.

Table 3 Deleted System/370 facilities.

Deleted or replaced facility	*Number of instructions*	
	Unprivileged	*Privileged*
System/370 channels	—	10
Channel-set switching	—	2
Segment protection	—	—
Basic-control mode	—	—
Interval timer	—	—
Direct control	—	2
External signals	—	—
64K-byte segment size	—	—
2K-byte page size	—	—
2K-byte protection-block size	—	3
Three signal-processor orders	—	—
Totals	0	17

The 370-XA interpretive-execution facility replaces the various VM assists available in System/370 with a single facility that is integrated within the 370-XA architecture. It is extensible and readily allows new real-machine facilities to be made available on a virtual machine.

31-Bit addressing

The process to introduce 31-bit addressing into the System/360-System/370 architecture started in the late 1960s [7], when the new program-status-word (PSW) and control-register formats were established. Although it was clear at the outset that an entire four-byte field would have to be allocated for the extended address, the decision subsequently had to be made whether to use 31-bit addresses or continue with the 32-bit format introduced on the Model 67. The 31-bit format was chosen so as to provide space (high-order bit position) for a control or escape bit within the four-byte address field. It was felt that the ability to address 4G bytes ($G = 2^{30}$) of storage with a 32-bit address instead of 2G bytes with a 31-bit address did not justify the potential inconvenience in the handling of the control or mode bits.

Since in System/370 the high-order byte was ignored in address generation and no programming discipline was enforced to ensure that it contained zeros, 31-bit and 24-bit addressing are incompatible. Therefore, 31-bit addressing is made available only in a special, program-controlled mode. The mode bit appears in bit position 32 of the PSW, with the instruction address in bit positions 33–63. The right half of the PSW thus takes on the form of an address constant, as shown in Fig. 1.

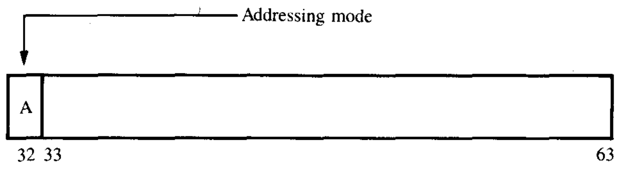

Figure 1 Format of the right half of the 370-XA program-status word.

In order to permit the gradual introduction of 31-bit addressing and to permit branching between modules using 24-bit and 31-bit addresses, a set of unprivileged instructions is provided for setting and saving a 31-bit address field and the mode bit.

The basic 370-XA calling (subroutine linkage) instruction intended for situations requiring the ability to switch the addressing mode is BRANCH AND SAVE AND SET MODE (BASSM). It saves the right half of the 370-XA PSW and sets the addressing mode as specified by the high-order bit of the branch address. The corresponding return instruction is BRANCH AND SET MODE (BSM), which omits the saving function. In order to permit saving the condition code and program mask, which was done by the System/370 BRANCH AND LINK but not by BASSM, a separate instruction, INSERT PROGRAM MASK, is introduced (it inserts these fields into the designated general register).

The two BRANCH AND SAVE (BAS and BASR) instructions, which were introduced recently in System/370, can be used for subroutine linkage in situations where the addressing mode does not change. They save the right half of the PSW but do not set the addressing mode. The System/370 BRANCH AND LINK (BAL and BALR) instructions are provided in the 370-XA architecture primarily for compatibility reasons. In 31-bit addressing mode, they are defined to operate the same as BAS and BASR. They have the advantage that they are available in both the System/370 and 370-XA architectures and hence are convenient for programs that must run in either architecture. Use of BAS and BASR is recommended, however, because these instructions arrange link information in an architecture-independent format and are likely to be faster.

As part of the extension of the logical address to 31 bits, the program-event-recording (PER) starting and ending addresses in control registers are extended to 31 bits. Since the PER addresses are set up by the control program and are used to monitor storage accesses by application programs that may switch between the 24- and 31-bit addressing modes, they are 31 bits long independently of the addressing mode.

As part of the extension of the real address to 31 bits, all real addresses in dynamic-address-translation tables, dual-address-space tables, the prefix register, and the associated table-origin designations in control registers are extended to 31 bits. Since these addresses are part of the system-control structure and must coexist with application programs operating in either the 24-bit or 31-bit mode, they, too, are independent of the addressing mode. This extension is one of the reasons, of course, why 370-XA is incompatible with System/370 for control programs.

The three new instructions for handling the storage key (INSERT STORAGE KEY EXTENDED, SET STORAGE KEY EXTENDED, and RESET REFERENCE BIT EXTENDED) were introduced into System/370 for operation with a 26-bit real address initially made available on the 3033 with the extended-real-addressing facility. Instead of extending the address in the original three storage-key instructions, new instructions were introduced at that time, partly for compatibility reasons but mostly because of implementation considerations on the 3033. The expansion of the address in a register from 24 to 26 or 31 bits for the two original RR-format instructions INSERT STORAGE KEY and SET STORAGE KEY would have been relatively easy, but the 3033 could not readily incorporate the change necessary to generate an address in excess of 24 bits for the S-format instruction RESET REFERENCE BIT. The new storage-key instructions were defined to operate on 4K-byte blocks instead of the original 2K-byte blocks; and, in anticipation of 370-XA, these instructions, as well as the new TEST BLOCK instruction, were defined to use 31-bit addresses. In the presence of the 370-XA addressing-mode control, however, these instructions in the 370-XA architecture are defined to operate with 24-bit or 31-bit addresses, as specified by the mode bit. This change was made for architectural consistency. The original three System/370 storage-key instructions are deleted in 370-XA.

The introduction of 31-bit addressing affected two fields in the System/370 channel architecture that are carried into 370-XA. The indirect-data-address word (IDAW) was extended to 31 bits with the introduction of 26-bit real addressing and was the only way at that time for the channel to access data beyond the 16M-byte limit. This format is now also used in 370-XA. Additionally, 370-XA introduces a new channel-command-word (CCW) format, called Format-1, with a 31-bit data-address field. The address formats used in I/O operations, of course, are not subject to the control of the addressing mode in the PSW.

Page protection

The page-protection facility controls whether storing is allowed in a page of virtual storage. It is controlled by a bit in the page-table entry associated with the DAT mechanism.

Page protection supersedes the segment-protection facility offered on the 3081 and 3083 CPUs in System/370 mode and offers the same function with a smaller block size.

The page-protection mechanism is in addition to the key-controlled and low-address protection mechanisms. Access to a storage location is permitted only when none of the protection mechanisms prohibit it. Key-controlled protection applies to physical storage and guards it against unauthorized storing or unauthorized storing and fetching by CPU and I/O. It was introduced in the original System/360 and, in the absence of DAT, was the primary mechanism for allocating main storage. Low-address protection, which was first made available in 1978 on the 3033, protects the contents of locations 0–511 against improper storing and is intended to guard system-control information against bugs in the operating system.

The page-protection facility permits establishing read-only pages in a user's virtual storage. By controlling access to sections of virtual storage by separate sets of segment and page tables, storing can be selectively controlled for any number of users and combinations of pages. For example, VM/370 uses the System/370 segment protection facility to establish read-only segments common to a number of users. One of the MVS uses of page protection is to guard against bugs in the control program and to preclude storing into certain pages that are associated with a storage key of zero.

Tracing

The 370-XA architecture introduces a new tracing facility as an aid in the determination of system problems. It includes three separately controllable functions that cause entries to be placed in the trace table designated by control register 12: branch tracing, address-space-number (ASN) tracing, and explicit tracing. When branch tracing is on, a trace entry is formed for the successful execution of BRANCH AND LINK (BALR), BRANCH AND SAVE (BASR), and BRANCH AND SAVE AND SET MODE (BASSM). When ASN tracing is on, each execution of PROGRAM CALL, PROGRAM TRANSFER, and SET SECONDARY ASN causes a trace entry to be formed. Explicit tracing is accomplished with the instruction TRACE, which forms an entry that includes bits 16–63 of the time-of-day clock, the designated storage operand, and the contents of the designated range of general registers.

The 370-XA trace facility differs in some significant ways from the facility that was initially introduced into System/370 in the form of an MVS assist and that was subsequently expanded as part of the dual-address-space facility on the 303X machines. The latter was a machine adaptation of the tracing as originally implemented in software in MVS, and in a multiprocessing system both CPUs placed entries into the same table, designated by the contents of a storage location. In a machine organization employing a cache for each CPU, such sharing introduced inefficiencies, as alternate storing of entries by the two CPUs required each CPU alternately to update the address at the common storage location, refetching it from main storage and purging it in the other CPU's cache. In 370-XA, each CPU has its own trace table.

The implicitly formed 370-XA trace-table entries consist of only one or two words and contain only the essential information for identifying an event, whereas in the System/370 trace each entry consisted of 32 bytes. This choice for 370-XA was made in view of the overall consideration that, typically, the trace information is needed to identify an unexpected error. The system normally would be expected to operate with tracing enabled and with the table wrapping around over itself. Upon detection of an error, the software would retrieve the last table's worth of entries in each CPU, merge them into a single system table, and expand the entries with the needed additional information. The additional software effort to expand and merge the "lean" tables private to each CPU is expected to be less than the CPU performance degradation that could be expected otherwise.

SIGNAL PROCESSOR extensions

The facilities for communication between CPUs in a multiprocessing system have been extended by incorporating in the SIGNAL PROCESSOR (SIGP) instruction an ability whereby certain orders can transfer a parameter to the designated CPU. Two new SIGP orders are added, both of which make use of this parameter: *set prefix*, which provides the capability to set a specific value in the prefix register of the designated CPU, and *store status at address*, for saving the CPU status of the designated CPU at a designated location. The purpose of these extensions is to eliminate the dependence of system operation on the availability of the first 4K locations of main storage and to reduce potential conflict in its use.

The System/370 SIGP restart order was defined to cause the same action as the manual restart key—storing the old PSW at real location 8 and fetching the new one from real location 0. Furthermore, the manually initiated initial-program-loading procedure also uses real location 0. Thus, in addition to requiring that storage location 0 be operational, the System/370 restart order shares location 0 with two other functions. Although contention was possible when manual restart was invoked at the instant location 0 had been set up for programmed restart, no conflicts were expected to occur in normal operation.

Two 370-XA extensions tend to aggravate potential conflicts in the use of location 0. In one mode of operation, the virtual machine is assigned absolute location 0, and the potential for contention increases because four-way multiprocessing is implemented. Furthermore, a similar conflict now exists in the use of the first 4K locations by the System/370 store-status function and a virtual machine.

The parameter associated with the two new SIGP orders eliminates the contention involving virtual machines and permits any 4K-byte block to be used for the SIGP restart and store-status functions. In a multiprocessing system comprising more than two CPUs, it permits one CPU to perform a store-status or restart function at another CPU while a third CPU is operating as a virtual machine.

DIVIDE instruction

The extended-precision floating-point facility, first shipped in 1969 on the Model 85, introduced the 128-bit extended-precision format, having a fraction that is equivalent to approximately 34 decimal digits. It included instructions for adding, subtracting, and multiplying 128-bit operands to yield a 128-bit result. Because of its relatively low frequency of use and the Model 85 design constraints, the facility did not include a DIVIDE instruction [8]. The architecture for DIVIDE, however, was defined, including the instruction format, operation code, mnemonics, and operation, and both the System/370 assembler and Fortran compiler treated the instruction as if it were available on the machine. During execution, the instruction caused an interruption for operation exception, and the function was simulated.

The System/370 extended architecture includes the extended-precision floating-point DIVIDE as originally defined and previously simulated. It divides a 128-bit dividend by a 128-bit divisor to yield a 128-bit quotient.

Conclusion

The 370-XA architecture has evolved from the System/370 architecture in response to two types of requirements—to permit improvements in the operating system structure and performance and to introduce new function for application program use. The constraints associated with these requirements differ significantly. In the case of operating-system functions, the extent of change to System/370 facilities was limited primarily by the cost of the associated changes in the operating system, whereas for facilities exposed to application programs, strict compatibility from the System/370 to the 370-XA architecture was expected.

Extensions have been incorporated to meet both types of requirements. Some facilities introduced for operating-system purposes have been additions to System/370 architecture, whereas for others the new facilities were significant but still localized replacements of the corresponding System/370 facilities. The 31-bit addressing facility introduced a major functional extension for application programs, while still permitting System/370 application programs to run on 370-XA machines.

The 370-XA architecture is the most significant extension to the original System/360 architecture since the introduction of System/370 in the early 1970s. It demonstrates that the basic System/360 design can be adapted to new requirements 18 years after its initial introduction.

Acknowledgments

Although many people participated in the development of the 370-XA architecture, the contributions of the following colleagues deserve special note. R. L. Cormier, R. J. Dugan, and R. R. Guyette had the technical responsibility for the architecture of the 370-XA channel subsystem, including the writing of the corresponding chapters of the *IBM 370-XA Principles of Operation.* P. H. Gum had the technical responsibility for the interpretive-execution architecture and participated in the development of other facilities. R. M. Smith made significant contributions to most aspects of CPU architecture and had the technical responsibility for the CPU part of the *IBM 370-XA Principles of Operation.*

References and notes

1. The term *architecture* is used here to describe the attributes of a system as seen by the programmer, that is, the conceptual structure and functional behavior, as distinct from the organization of the data flow and controls, the logical design, and the physical implementation.
2. *IBM 4300 Processors Principles of Operation for ECPS:VSE Mode*, Order No. GA22-7070, available through IBM branch offices.
3. *IBM 370-XA Principles of Operation*, Order No. SA22-7085, available through IBM branch offices.
4. R. L. Cormier, R. J. Dugan, and R. R. Guyette, "System/370 Extended Architecture: The Channel Subsystem," *IBM J. Res. Develop.* **27**, 206–218 (1983, this issue).
5. *IBM System/370 Principles of Operation*, Order No. GA22-7000, available through IBM branch offices.
6. The MOVE INVERSE instruction is listed in the *IBM System/370 Principles of Operation* since it is offered in System/370 mode on the 4331 and 4341 processors; it does not appear in the *IBM 370-XA Principles of Operation* because no machine implementing this architecture offers MOVE INVERSE.
7. R. P. Case and A. Padegs, "Architecture of the IBM System/370," *Commun. ACM* **21**, 73–96 (1978).
8. A. Padegs, "Structural Aspects of the System/360 Model 85; Part III—Extension to Floating-Point Architecture," *IBM Syst. J.* **7**, 22–29 (1968).

Received September 3, 1982; revised December 22, 1982

Andris Padegs *IBM Information Systems and Technology Group, P.O. Box 390, Poughkeepsie, New York 12602.* Dr. Padegs is the manager of the Central Systems Architecture Department, with responsibility for System/370 input/output and central-processor architectures and their extensions. He joined IBM in Poughkeepsie in 1958. After participating in the planning of the Stretch system, he was involved in developing channel architecture for System/360. In 1968 he became manager of central-processing-unit architecture

for System/370, and subsequently has managed the development of other architectures. Dr. Padegs received an A.B. degree (*magna cum laude*) from Dartmouth College, Hanover, New Hampshire, in 1953, an M.S. in electrical engineering from Thayer School of Engineering in 1954, and a Ph.D. in electrical engineering from Carnegie Institute of Technology, Pittsburgh, Pennsylvania, in 1958. He received an IBM Outstanding Contribution Award in 1964 for his contribution to System/360 architecture, an IBM Outstanding Invention Award in 1966 for development of the System/360 input/output interface, an IBM Outstanding Contribution Award in 1973 for his work on System/370 architecture, an Invention Achievement Award in 1974, and a Division Award in 1981 for work on System/370 extensions. Dr. Padegs is a member of Sigma Xi and the Institute of Electrical and Electronics Engineers.