A. Padegs

System/360 and Beyond

The evolution of modern large-scale computer architecture within IBM is described, starting with the announcement of System/360 in 1964 and covering the latest extensions to System/370. Emphasis is placed on key attributes and on the motivation for providing them, and an assessment is made of the experience gained in the implementation and use of the architecture. The main approaches are discussed for obtaining implementations at widely differing performance levels, and a number of significant implementation parameters for all processors are listed.

Introduction

With the introduction of System/360 in 1964, a major change in the development of computers within IBM took place. With the recognition that architecture [1] and implementation could be separated and that one need not imply the other, a common machine architecture was established. It was intended for program-compatible embodiments over a wide range of performance levels and for various types of applications.

Since its introduction, this architecture has been the basis for all intermediate and large computers produced by IBM. It has also become the basis for machines produced by a number of other manufacturers in the USA, Japan, and the Soviet Union. The architecture has provided a firm interface for application development, and it has permitted the operating systems to grow significantly in size and function. Although the architecture was developed when logic technology with a single device per chip and magnetic cores were used to implement machines, its fundamental structure is still suitable for today's designs, which use dense arrays and integrated circuits of thousands of elements per chip.

This paper reviews the salient characteristics of the System/360 architecture and its follow-on, the System/370 architecture. The objectives are to cover significant accomplishments, to give the motivation for key architectural decisions, and to present an assessment; it is not intended that the paper provide a complete historical record of IBM's architecture developments. Furthermore, the paper does not contain a detailed technical review of design considerations and alternatives; this type of review has been published previously [2–9].

The first two sections review the System/360 and System/370 architectures, stating the objectives, constraints, and contributions. Following this, developments in I/O architecture are outlined. The introduction of various enhancements to the architecture is discussed in relation to product announcements; and, in a separate section, the significance of microprogramming and the cache to the implementation of a compatible line of machines is reviewed. The final sections are devoted to an assessment of IBM's experience with System/360 and System/370. In an appendix, tables comparing implementation characteristics for all processors provide further illustration of the steps taken to achieve different performance levels.

System/360 architecture

System/360 was the result of a major effort to design an architecture for a new line of computers that was unencumbered by the requirement to be compatible with existing architectures. Work on a new architecture for a family of machines began in the early 1960s; its specifications were released in April 1964, when the first models of System/360 were announced.

When System/360 development was initiated, most new computer models were, from the viewpoint of their logical structure, improved, enlarged, or technologically recast versions of the machines developed in the early

1950s. IBM products had evolved from 701 to 7094 II, from 702 to 7080, from 650 to 7074, and from 1401 to 7010 [10]. Additionally, IBM had produced Stretch, formally known as the IBM 7030; it had been developed largely as a project to challenge the state of the art, but from the point of view of architecture it was a predecessor of System/360.

In many ways the design concepts underlying System/360 [2–5] were the same as those for Stretch [11]. Both Stretch and System/360 provided, in a single architecture, facilities suitable for scientific, commercial, and real-time applications; both placed major emphasis on the generality and code-independence of instruction and data formats; and both provided for the uniform attachment and control of a wide variety of I/O equipment. System/360, however, was the first demonstration that the concept of developing an architecture for a family of compatible machines was practical, with the initial implementations targeted to yield models with internal performances ranging from that of the IBM 1401 to well beyond that of Stretch. Intermodel compatibility was probably the most far-reaching requirement in developing System/360 and one which affected both the architecture and the procedures for developing and controlling it.

System/360 incorporated a number of the new architecture concepts introduced in Stretch, such as binary storage subdivision and the eight-bit byte, storage protection, and a generalized interruption mechanism. However, the System/360 architectural definition of some of these concepts differed from that in Stretch because the environment and objectives were different. Since Stretch had the flavor of an experimental computer, its architecture could afford to strive for a set of functions of great logical consistency and completeness. System/360, on the other hand, was defined at its inception as a base for a product line. As such, its development called for a more frugal choice in the selection of functions, based on a critical evaluation of the available experience. The architecture had to encompass implementations covering wide performance and cost ranges, and its definition had to reflect compromises between the performance and cost objectives of large and small machines. As a result, Stretch innovations such as storage addressing to the bit level, variable byte size, and automatic handling of floating-point range exceptions were not included.

The following are the key areas where System/360 introduced innovations or otherwise determined direction in architecture.

Addressing For efficiency and because of the ease of table utilization, System/360 uses binary-radix storage addressing, with 24-bit addresses that designate byte locations. Thus System/360 can address 16M eight-bit bytes, as compared to the 2M bytes on Stretch (M stands for 2^{20} or 1 048 576). This addressing capability is, of course, available on all models and should be considered in light of the maximum storage sizes available at that time: 32K 36-bit words on the 7094 II (K stands for 2^{10} or 1024), 160 000 six-bit characters on the 7080, 300 000 digits on the 7074, 100 000 seven-bit characters on the 7010, and 16 000 seven-bit characters on the 1460 (the word-mark bit is included in the 7010 and 1460 character sizes). The smallest initial System/360 model, Model 30, offered up to 64K eight-bit bytes, and 1M bytes was available on the largest initial model, Model 75. The ability to address and to effectively utilize large storage was one of the key attributes of System/360.

Address generation A truncated 12-bit address in an instruction, in conjunction with a full base-address value in a register, provides indexing and eliminates the inefficiency of carrying a full address in each instruction. A second level of indexing is available in some instructions to facilitate loop control. The 12-bit displacement, without an index or base value, provides addressability for loading or saving the base values but normally is so small that all programs need base addresses and are thus generally location-independent.

Provision for control program A comprehensive interruption system, supervisor and problem states, storage protection, and an interval timer provide a basis for designing a secure operating system. Input/output instructions are invalid in the problem state, and means are provided for the supervisor to control the duration of application programs and switching between them. (In Stretch an operating system could be modified by unauthorized input from an I/O device.)

Input and output The multiplexer channel and a common method of attaching and programming all I/O devices extended the concepts introduced in 702 and Stretch. These aspects are discussed subsequently in the section "Input and Output."

General-purpose registers Sixteen registers serve as accumulators for fixed-point and logical operations, as well as sources of base and index values in address generation, thus bringing the full power of the fixed-point arithmetic-operation set to bear upon indexing computations.

Character size In contrast to the straight six-bit approach used in the IBM 702–7080 and 1401–7010 families, two character sizes were introduced: eight-bit codes for alphanumeric, and four-bit codes for numeric characters. This approach, used in the IBM 650–7074 family, has

greater coding efficiency, with spare code points in the alphabetic set, and is commensurate with binary subdivisions used in the rest of the system. The length of operands is specified in the instruction: Decimal operands can be up to 16 bytes in length; character operands are variable up to 256 bytes. In Stretch any character size from one to eight bits could be specified, but variable-field-length operands were limited to 64 bits.

Floating-point data format Two formats were introduced, both available on all models: a 64-bit format for use in precision-sensitive problems and a 32-bit format for faster speed and conservation of storage space. The 32-bit format was intended primarily for the smaller models, where differences in the execution time for the two formats were significant. The alternative would have been a single 48-bit format to succeed the 36-bit format of the 7094 and the 64-bit format of Stretch. Both the 32-bit and 64-bit formats use the same exponent size, with a base of 16. This was a departure from base 2 and was introduced to permit simpler circuitry and to reduce the frequency of pre-shift, overflow, and precision-loss post-shift in addition and subtraction [12].

Serviceability The ability to automatically record the detailed machine state at the instant of an error and to initialize it to any specified value provides tools for significant serviceability improvements [13].

The models at both ends of the performance range introduced architecture changes to meet their particular cost and performance goals. Model 20, although nominally called part of the System/360 family, was incompatible with System/360. Its architecture provided for a maximum main storage of 64K bytes, and it had 37 instead of System/360's 143 instructions. Other differences were that it did not include the supervisor state, had its own set of I/O instructions, and had a 32-bit instead of the 64-bit program-status word (PSW). Compatibility for running application programs was affected because the Model 20 omitted floating-point and 32-bit binary arithmetic, had eight 16-bit instead of sixteen 32-bit general registers, and had a special direct-addressing mode for forming storage-operand addresses.

At the other end of the performance range, the Models 91, 95, and 195 introduced some deviations to accommodate their highly overlapped designs by delaying program interruptions and permitting the result of a divide operation to be off by one bit in the low-order bit position. These differences required some adjustments in software but did not affect compatibility for application programs in any significant way.

Additional functions were introduced by a few of the later System/360 models. Model 44, announced in 1965, had a number of extensions for real-time applications, which, however, were not continued in later models. At the same time, Model 67 introduced virtual storage [14], which, with some modifications, became a basic part of System/370. The 9020 System, which was developed for the Federal Aviation Administration, interconnected modified Model 50s into a multiprocessing system to meet exceptionally stringent requirements for continuity in machine operations [15]. Models 65 and 67 both offered multiprocessing facilities [6], which later were significantly extended for System/370.

Then, in 1968, as part of the extended-precision floating-point facility on the Model 85, the 128-bit floating-point format was introduced [7]. The package also included special instructions for rounding floating-point numbers when going to a shorter format; they alleviated somewhat the lack of rounding in the original architecture. Model 85 also removed the original System/360 requirement that storage operands for unprivileged instructions be aligned on boundaries equal to a multiple of the operand length. Both of these extensions were carried into System/370. Additionally, the 2880 Block-Multiplexer Channel, on System/360 Models 85 and 195, had many of the System/370 I/O architecture extensions [8].

In two areas the original System/360 decisions on user-oriented functions were subsequently changed. The first one concerns floating-point instructions. The original System/360 architecture did not anticipate the significance of compatibility in the handling of overflow and the need for indicating the true result value on both overflow and underflow. Furthermore, it had overlooked the need for a guard digit in post-normalization. Both of these functions were changed in 1968, with all installed machines retrofitted.

The other change concerns the encoding of decimal data. System/360 anticipated the adoption of a proposal for a seven-bit American Standard Code for Information Interchange (ASCII) and of a technique for expanding the seven-bit code to eight bits. It provided a mode bit in the PSW that specified, for the code-sensitive instructions, operation with either the ASCII or the Extended-Binary-Coded-Decimal-Interchange Code (EBCDIC). The eight-bit extension of the code was never adopted as a national standard, and the ASCII mode has subsequently been deleted in the System/370 architecture. It was highly unlikely that any production programs ever used the eight-bit ASCII code; none have ever been identified. The mode bit subsequently turned out to be the only unassigned one in the PSW and was convenient for distinguishing between the original and the extended-control (EC)-mode PSW format introduced for System/370.

In a number of other areas, different architectural choices, in retrospect, might have been preferable. The problem of storage-address size is discussed later in the paper. For some areas, extensions have subsequently been introduced to solve the constraints set by the initial decisions; for example, the new EC-mode PSW format corrected the lack of extensibility in the original PSW format, and the early release of the CPU during execution of the new System/370 START I/O FAST RELEASE instruction eliminated the unnecessary delay required by the original START I/O in high-performance machines. In other areas, the need to preserve compatibility has been felt to be so overwhelming that the reevaluation of the original architectural choices has been of no practical interest. This applies particularly to problem-program functions, such as whether the floating-point significance exception is really useful, and whether the EDIT and EDIT AND MARK instructions are warranted in view of their very specific operand formats.

System/370 architecture

In contrast to System/360, the objective of System/370 was an evolutionary extension of System/360 architecture for a new set of models and for new releases of programming systems [9]. Experience with the System/360 architecture had identified a number of bottlenecks and limitations in the efficiency of system use and had pointed out areas where additional machine functions were desirable. Furthermore, because the cost of technology for main storage and logic circuitry was becoming lower in relation to the overall system cost, it was feasible to consider extending the machine architecture; it was possible, in fact, to economically include functions that did not appear justified when System/360 was developed. For example, because of cost considerations in the smaller models, System/360 provided only one 32-bit timer in a main-storage location, which had to be programmed to provide all timing functions. System/370 introduced three distinct facilities, each with a 64-bit value: a time-of-day clock (for real-time indication), a clock comparator (a real-time alarm clock), and a CPU timer (for measuring process time) [9].

System/370 was constrained to be upward-compatible for System/360 application programs and for the main-line operating systems. Even though such operating systems could not benefit from the new functions available in System/370, and new support was planned, the ability to run those operating systems was needed for the transition period. For this reason, System/370 continued to provide functions, such as the System/360 timer and the System/360 PSW format, that had in fact been superseded by functionally richer extensions. Additionally, System/370 needed to attach and operate System/360 I/O devices.

System/370 evolved, and its architecture [16] was released, in a number of increments. The system was introduced in June 1970 with the announcement of Models 155 and 165, at which time the main architectural extensions were six general-purpose instructions, the time-of-day clock (with a period of 143 years and a resolution of one microsecond), and control registers (they serve as an extension of the PSW). The original System/370 also included a number of extensions to enhance model-independent recovery by software from machine malfunctions [17]. Virtual storage, the CPU timer, the clock comparator, program-event recording (for software debugging), and the new PSW format and interruption controls associated with the extended-control (EC) mode were introduced with the announcement of Models 158 and 168 in August 1972. Multiprocessing and the conditional-swapping and PSW-key-handling instructions were introduced in February 1973.

Then, with the introduction of the 3033, a number of extensions were made available that enhanced the performance and function of the MVS operating system. One-level addressing and the instruction MOVE INVERSE were introduced as part of the VSE (virtual storage extended) mode on the 4300 processors to meet the needs of the DOS/VSE operating system [18, 19]. The MOVE INVERSE instruction is intended primarily for environments, such as the Arabic language, where text is arranged in a right-to-left order.

The single item that most distinguishes System/370 from System/360 is the availability of a dynamic-address-translation facility, which allows the control program to efficiently implement a group of functions collectively referred to as *virtual storage*. The approach incorporates *paging* from external storage as introduced in Atlas [20] and a second level of indirection, *segmentation*, as suggested by Dennis [21] and as further detailed by Arden *et al.* [22].

The System/370 version of this facility is largely patterned after the System/360 Model 67 [14]. Experience with that machine and its operating system, TSS, had verified the value of many of its concepts and had provided actual usage data for making System/370 design decisions. In addition to a number of format changes, System/370 offers two page and segment sizes to accommodate both large and small systems, but it does not offer 32-bit virtual addressing, which was available on the Model 67. The System/370 virtual-storage operating systems were evolutions of the corresponding real-storage operating systems and could not accommodate 32-bit addresses.

The virtual-storage architecture of the 3033 and other large processors was enhanced early in 1981 by the introduction of the dual-address-space facility. This extension includes a 16-bit address-space number, which is associated with a set of segment and page tables and identifies a virtual address space of 2^{24} bytes. A total of 2^{16} address spaces can be established, although at any one time addressability exists to two address spaces—the primary and secondary. Instructions and controls are provided to load an address-space number so as to establish addressability, call and return from programs in either the same or another space, move data between spaces, and establish authorization for these operations. These facilities extend the size of the addressable virtual storage and provide a basis for enhancing system integrity.

In the VSE mode, the main change was the substitution of the *one-level-addressing* facility for the System/370 dynamic-address-translation facility. DOS/VSE offers one virtual address space of up to 16M bytes, and the architecture is simplified accordingly by eliminating the multiple-address-space capability of the System/370. Storage is directly addressable by the CPU and all channels, using a uniform set of virtual addresses. The translation table is in internal machine storage, and special instructions are provided for setting up the mapping. Protection, by means of storage keys, applies to virtual instead of real storage. Because of the simpler translation procedure and the ability of channels to use virtual addresses, performance gains are possible, and the software for translating addresses in channel programs is eliminated. The VSE mode is compatible with System/370 for problem programs, but not for the control program. The full System/370 facilities are available on the 4300 processors in the System/370 mode.

Another major functional extension is the inclusion of a number of facilities that permit formation of a *multiprocessing* system, where two or more CPUs share common main storage and are controlled by a single copy of the operating system. The concept of using a prefix to offset the main-storage address when accessing the block containing shared control information is the same as that used in System/360. The architecture was extended, however, by making the prefix settable by the program (instead of manually) and by providing the SIGNAL PROCESSOR instruction and a special interface for communicating between CPUs. On the 3033 a further extension made it possible for the software to connect a set of channels to one of two CPUs.

The main extension to the multiprocessing architecture, though, was in the control of accesses to shared main storage. In a multiprocessing system, the conventions of a uniprocessor communication protocol become inadequate when one CPU is changing the contents of a common storage location while the other is observing it, or when both CPUs are updating the contents of the location at the same time. The System/370 architecture includes a number of rules on the concurrency, multiplicity, and order of storage accesses, and specific instructions are introduced to permit sharing of serially reusable resources, such as updating chained lists. Specifically, in System/360, the TEST AND SET instruction provided a means whereby the inspection of a bit in storage and the setting of it to one could be performed indivisibly. In System/370, the two compare-and-swap instructions indivisibly compare a field in storage with a value in a register and, upon matching, replace the storage operand with a new value [16].

The *IBM System/370 Principles of Operation* [16] in the Spring 1981 edition contained a total of 204 instructions, as compared to the 143 initially available in System/360 [23]; this provides one indication of the growth of the architecture. Of the 61 new instructions, 39 are either privileged or semiprivileged (11 of the original System/360 instructions were privileged), indicating that a relatively larger portion of the architectural extensions is intended for system functions.

Input and output

The concept of a common method of I/O attachment and control evolved gradually. The 702 had a common interface for attaching I/O control units and a common architecture for controlling I/O operations. The 709 introduced the concept of a channel. The Stretch "exchange" [10] provided a mechanism for sharing equipment for multiple I/O operations, using a common I/O interface and I/O architecture (a different interface was used for attaching the disk unit to the high-speed exchange, and the instructions for its control differed somewhat). In 1961 a standard interface was established for attaching I/O to all new large systems. It was a modification of the Stretch interface and was available on the IBM 1410/7010, 7040/44, 7070/74, 7080, and 7090/7094 systems for attaching disk, magnetic tape, and communications control units.

System/360 extended the standardization of I/O attachment and control by applying a common attachment interface and a uniform program control to a larger variety of device types and covering a wider spread of data rates.

- *Channels*

System/360 introduced the *subchannel*, which for most purposes gives the appearance of an independently oper-

ating processor that can sustain its own channel program. Different types of channels were designed, and, depending on the type of channel, different levels of concurrency among channel programs were made possible. A *selector* channel has one subchannel and permits operation with one device at a time, normally at a high data rate. A *multiplexer* channel can have up to 256 subchannels and, conceptually, can be executing a channel program for each subchannel. The actual level of interleaving depends on the type of multiplexer channel and the device. A *byte-multiplexer* channel is designed for low-speed operation to interleave individual bytes or bursts of bytes from such devices as keyboards, communications lines, printers, and card equipment. When it was introduced, it represented a major advance for communications-based systems and real-time applications [5]. The *block-multiplexer* channel, introduced later for System/360 Model 85, is intended for high-speed operation and is particularly advantageous for use with rotating storage devices, such as disks and drums [8]. When used in conjunction with rotational-position sensing, it permits a subchannel to be assigned and a channel program to be established for each access arm, with each program monopolizing the channel for the duration of data transfer but releasing channel facilities during arm movement and during the rotational delay associated with locating the designated record.

- *I/O interface*

The *System/360 I/O interface* is the connection between a channel and an I/O control unit; it provides the necessary physical, electrical, and communications-protocol specifications. It is based on the standard interface of 1961.

The original System/360 I/O interface specification was adequate for data rates up to about 1M bytes per second for a cable length of about 100 feet. For cable length of the order of 20 feet, the IBM 2301 Drum Storage, with a rate of 1.2M bytes per second, could be accommodated. The fully interlocked signaling protocol allowed one channel-cable connection to sustain data transfer over a very wide range of rates, with both the channel and device having complete control of the timing of each byte transfer. It did, however, require an electrical signal to be propagated between the channel and the control unit four times for each byte transferred.

With the advent of auxiliary-storage technologies employing higher recording densities, it was necessary to increase the data-transfer capacity of the interface. For some buffered devices a higher data rate was desirable to reduce the transfer time. Furthermore, many installations needed longer cable connections. To meet these goals, System/370 introduced changes both in the signaling protocol and in the width of the interface.

The System/370 I/O interface [24] includes two additional tag lines to provide the same level of transfer interlocks with only two propagation times per byte transferred. It depends on the control unit whether or not the new facility is used; thus, control units implemented to operate with the System/360 protocols can be attached to System/370 channels. On some System/370 channels and control units the bus width can be extended optionally to two bytes, thus doubling its data-transfer capacity.

As a result of these two additions, the System/370 I/O interface can sustain a data-transfer rate of over 1.5M bytes per second in the one-byte version and over 3M bytes per second in the two-byte version, over a cable length somewhat less than 100 feet; longer distances can be accommodated at lower data rates.

The *data-streaming* mode, introduced recently for the IBM 3380 Disk Storage, eliminates the interlocks between the request and response signals during data transfer. Data, with the appropriate tag signals, are sent in the form of fixed-length pulses. This eliminates the dependency of the data rate on cable length caused by the interface protocol. The IBM 3380 specifications provide for a transfer rate of 3M bytes per second over 400 feet with the one-byte interface.

System/360 was the first system in which a common attachment interface was used to connect a large variety of I/O control units to a line of computers. The interface has been successful in a number of ways. It has offered an unprecedented choice of I/O equipment in configuring a system. It has permitted channels and control units to be designed independently and at different locations with an assurance that, assembled into a system on the user's premises, they will operate without any adjustments. Furthermore, the specific interface definition has been sufficiently general and flexible to accommodate new device types and to permit extension of function and data rates in a compatible manner. As a result, a control unit designed to the original definition (after the 1967 change to the electrical specifications) can operate with a channel incorporating the latest extensions, provided the channel meets the speed requirements.

The standard interface permits other attachment approaches. At the penalty of losing some configuration flexibility, the total cost of a system can be reduced by eliminating a separate frame and power supply for the control unit, by eliminating the use of an interface cable and the associated drivers and receivers, and by sharing some main-frame logic circuits for the control-unit functions. Such integrated designs are offered on the smaller System/360 and System/370 models for some common I/O device types. Even though such designs physically merge

the channel and control unit, they nevertheless maintain the logical separation and simulate those aspects of the standard interface that are observable by the program. Thus, regardless of the implementation, all I/O devices are controlled by the same set of I/O instructions, command words, and other program formats.

Implementation approaches

The various levels of performance and cost in the implementation of the architecture are achieved by appropriate choices and tradeoffs among such parameters as circuit speed and cycle time, width of data and logic paths, overlap of instruction execution, and speed, width, and interleaving of main storage [25, 26]. Two new developments in machine implementation, however, are particularly significant in the adoption and subsequent extension of System/360 architecture: microprogramming and the cache (high-speed buffer).

- *Microprogramming*

Microprogramming, originally suggested by Wilkes [27], is the use of simple and fast low-level instructions for controlling machine sequences [28-30]. This type of design permits sharing a basic data flow for a wide variety of functions and readily permits tradeoffs between cost and performance. With conventional logic circuitry, the cost of controls increases in a roughly linear relationship to the functional capability. With microprogramming, a base cost for the microcode-storage device and the supporting logic must be borne, after which the incremental cost for adding more storage in order to microprogram additional function is relatively small.

It was largely because of microprogramming and the economy associated with sharing hardware that it became economically feasible to implement the full System/360 architecture on the smaller models. The savings were particularly significant in the implementation of input and output, as microprogramming made it possible to build integrated channels where the logic capability of the machine is time-shared between CPU and channel functions. In such an implementation, the channel becomes a conceptual entity, and one may include a large number of subchannels at virtually no cost other than the storage space for the governing control information.

Microprogramming made it possible to incorporate in System/360 and System/370 models the capabilities for emulating other architectures, such as those of the IBM 1401 and the IBM 7094 [31]. It also made it possible to extend the original System/360 architecture with assists for specific operating systems. Furthermore, microprogramming has had a beneficial effect on the architecture-resolution process, since it permits corrections and changes in machine functions after the circuitry has been designed and built; some changes are feasible even after the machine has been delivered to the customer.

Microprogramming is used to varying extent in all System/360 and System/370 models except for Models 44, 75, 91, 95, and 195. The extent and the method of use depend on performance objectives. Larger models normally have more bits per microprogram-instruction word for the control of their more complex data paths. On the other hand, smaller models have larger microprograms, since these models require more cycles to accomplish the same function and use microprogramming for more functions. As an example, the Model 168 has 4K words of 108 bits each, whereas the Model 138 has 64K words of 18 bits each. In the initial System/360 models, microprograms resided in read-only storage, but in most later models read-write storage is used. In the smaller models, microprograms reside in an extension of main storage.

- *Cache*

Starting with System/360 Model 85, the larger models use a high-speed buffer, called the *cache*, for accesses to main storage. Although the concept had been considered previously [32], IBM was the first to implement a large cache in a commercial computer [33-36]. The cache was a major advance in system organization and subsequently has been extensively analyzed in the literature [37-39]. The cache is interposed between the CPU and main storage, and its existence is not apparent to the program.

The cache reduces the number of main-storage references, because information fetched into the cache can be reused without access to main storage. Furthermore, by loading entire "lines" (typically 32-64 bytes) on any request for storage information, the machine can prefetch valuable information for future use and thus avoid the delay associated with additional storage access. The effectiveness of the cache depends on its size and other design parameters, as well as on the distribution of addresses used to access storage. According to Liptay [34], on the Model 85 with a 16K-byte cache, typically 97% of fetches were satisfied with data from the cache. With larger caches, in scientific applications "hit" ratios of 99% and over can be attained, although for interactive environments a more typical ratio is 96%. Furthermore, by allowing channels to communicate directly with main storage, the cache reduces storage interference and improves accessibility of storage for I/O, thus permitting higher I/O data rates.

The effect of a storage hierarchy using a cache is to reduce the dependence of CPU operations on storage access time and to provide a better match between the

operation speeds of main storage and CPU circuitry. The cache provides more freedom in the choice of storage technologies and allows for larger storage and longer access times. The introduction of a cache played a significant role in the realization of systems with large main storage.

Experience with System/360 and its extensions

The following are some of the major observations to be made and conclusions to be drawn concerning System/360 and its extensions.

- *Implementation of compatible machines*

Experience clearly verified that the initial System/360 goals for a compatible line of machines were realistic, and that it was feasible to build a family of machines within which programs could be transferred routinely from one model to another. The validity of the original compatibility goals was particularly proven by the fact that other manufacturers have been successful in producing System/370-compatible machines. In fact, compatibility helped reduce development costs within IBM. The original System/360 plan called for verifying each element of software on each model. Because of the growing confidence that programs which ran on one model would also run on other models, it was possible to significantly reduce the amount of cross-verification performed.

The original System/360 announcement included processors with a performance range of 25 to 1. Six years later this had increased to around 200 to 1, and today the performance of the 3081 is approximately 450 times that of the System/360 Model 25.

- *Main storage*

Main-storage sizes grew more rapidly than was anticipated in the 1960s; the technological improvements, which reduced the cost, had occurred at a faster rate than was expected. Thus, it became obvious at the time System/370 was in the planning stages that the 24-bit main-storage address size would have to be extended eventually.

The extension of the address size, however, proved to be more difficult than first expected. The basic addressing mechanism of System/360 was well suited to extension, since it depended on base registers that were already 32 bits wide. The interruption mechanism and the I/O control formats, however, did not have the required extensibility, since immediate cost and performance consequences in 1962 had outweighed the need to meet eventual long-term requirements. More importantly, operating systems and compiler-produced application programs had used the extra bit positions in address words for control purposes and hence required extensive modification.

In all new formats introduced for System/370, such as the control registers and the EC-mode PSW, main-storage-address fields are assigned 32 bit positions, should they be needed for address expansion. On the 3033 and other large machines, however, real storage in excess of 16M bytes is accommodated by making use of unused bit positions in the translation tables.

- *Precision vs. unpredictability*

In order to ensure compatible implementations, the architecture has to be complete in that it must cover all functions of the machine that are observable by the program, including all the unlikely concurrent occurrences of different unusual exceptions. It either must specify the action the machine performs or state that the action is unpredictable.

Identical action in all machines is less likely to cause problems with compatibility and has a certain aesthetic appeal. Indiscriminately specifying predictable operation, however, may present problems when the predictable operation is of insignificant value to the user and some later machine has difficulty complying with the required predictability. Whereas specifying initially that an operation is unpredictable might have been quite acceptable, relaxing the architecture definition to permit unpredictability has certain risks, because some programs may have come to depend on the initial, precise definition. Thus the architect has to make a deliberate decision about the extent of predictability.

The System/360 architecture did not provide adequate precision and detail in some areas. Because there was no specification of the priority in which concurrently existing program exceptions are recognized, programming of virtual machines was made difficult. Because the sequence and concurrency for storage accesses were not specified, processors could not communicate reliably using shared main storage. And because not enough details in machine-check handling were specified, the possibility of model-independent recovery after an equipment failure was reduced. The 1973 edition of the System/370 definition was more detailed and precise, but, for the sake of simplicity of the architectural model, specified as predictable some aspects that, as experience indicated, should not have been. An appendix in the 1980 edition of the *System/370 Principles of Operation* [16] lists six changes where the requirements for predictability have been relaxed. These changes concern such aspects as indicating an access exception for an operand when the instruction can be completed without the use of the operand, and they are unlikely to affect any program.

• *Assists*

In addition to the general-purpose architecture included in the *Principles of Operation*, many CPUs include special-purpose functions to improve the performance of a specific programming system. These functions, referred to as *assists*, comprise frequently occurring instruction sequences of a particular application, and a single operation code (or the occurrence of some other condition) may invoke the execution of an extensive procedure.

The assists are made possible by microprogramming and are implemented mostly (and in most machines exclusively) in microcode. They are particularly effective in improving performance when the function includes an interruption sequence and the associated program action; for example, when operating under VM/370, depending on the model and the operating system, a 40–65% reduction in elapsed time due to the VM assist has been measured [40].

The assists, however, are temporary internal interfaces and are not intended for application-program development. The functions may change between releases of the operating system, and, since the design decisions may be made on the basis of tradeoffs involving only a few specific machines, they may vary between models.

• *Levels of compatibility*

With the establishment of the operating system as an essential component of a user's installation, part of the architected machine interface is becoming an internal interface between the machine and the operating system. The dynamic-address-translation mechanism and machine-check indications are some examples of functions that do not directly affect the user, but the operating-system-dependent nature of the interface is particularly emphasized by the introduction of the assists. Furthermore, since the larger models normally are used with functionally richer operating systems and since the smaller models are usually restricted to those with lower storage requirements, an affinity has developed between machine power and operating-system power. Because of the nature of this affinity, it is not essential that the part of the machine interface affecting only the operating system be the same on all machines.

This evolution points out that two types of requirements for compatibility have to be considered. In order that old application programs run on new machines, the machine, jointly with the system program, must ensure that the basic facilities intended for *application-program* development continue to be available. On the other hand, changes may be acceptable in those facilities that are available to and affect only a *system program* or that can be masked by the system program from the application program. Indeed, such changes have to be expected, since they make it possible to improve the performance of system functions.

Such changes (as contrasted to extensions) have been introduced at different times into the System/360 and System/370 architectures. In the VSE mode on the 4300 processors, the one-level-addressing facility replaced the dynamic-address-translation facility in order to improve virtual-storage management for small systems; it affected only the interface between the machine and the DOS/VSE operating system that uses it. Similarly, it was feasible to phase in the extended-control (EC) mode, with the associated changes in interruption control and the PSW format, since the machine format affected only parts of the control program. The OS/VS2 operating system, however, continued for years to maintain the original PSW format in areas where the format was exposed to the user, such as in the trace information.

Architecture control

The design of a compatible line of machines required a strict separation of the architecture and machine-design functions and the introduction of methodology for the control of architecture. One of the major effects of System/360 was to establish architecture as an autonomous function and to introduce the management tools, discipline, and procedures for adopting and controlling architecture [9].

Recognizing that any differences in wording may imply differences in function, consistency is achieved by having only one specification of the architecture; it tells IBM machine designers the functions the machine must provide, and it describes to IBM programmers how the machine operates. The same specification is made available outside IBM as the *Principles of Operation* [16, 18], and is the only authoritative specification that describes the architecture. An analogous specification exists for the I/O interface [24].

A set of procedures have been established for the development of an architecture, starting with the conception of the idea and ending with the formal adoption of a definition. These procedures provide for the assessment of the cost and value of a function and for the approval of the architecture by machine and software implementers. Rules have been established about the extent of architectural compatibility [16], and provision is made for deviating from the common definition.

Although the implementation of a line of compatible computers did not take an undue amount of effort, the design and control of architecture proved to require more attention to detail than originally anticipated. Furthermore, experience with System/360 and its subsequent extensions has shown that the management of architecture must be an ongoing operation to ensure a consistent technical interpretation and to ensure that the evolution of the architecture structure is governed by a consistent set of principles and a design philosophy.

Conclusion

System/360 architecture has provided the basis for a number of machine generations, and it has been able to evolve to respond to new technologies, programming-system structures, and user requirements. This has been possible because of the soundness of its basic structure, the rigorousness of its definition, and the recognition of the autonomy of the architecture function.

As machine, software, and system-design technologies advance, further evolution of the architecture is inevitable. Changes will be made to better meet user needs and to allow more efficient design of machines and their associated programming systems. Because of the magnitude of the investment in System/370 architecture, however, it will be even more essential to ensure compatibility with the current architecture for those interfaces that are exposed to the user and are intended for application-program development.

Appendix: Model characteristics

This appendix summarizes some attributes of IBM machines implementing System/360 and System/370 architecture. Only the most recent characteristics are listed; some of the models were improved after initial announcement. The tables in this section are updated and extended versions of those published by Case and Padegs [9], and some corrections have been included.

Table 1 (appearing on pages 388–389) lists some key characteristics of the CPU and storage. CPU data-flow width indicates the largest field that can be handled in one cycle time. Depending on the CPU, a different amount of "work" is accomplished per CPU cycle; hence the cycle time cannot be used directly as a measure of relative speed.

Control storage contains the microprogram. A range in the size is given for those models where the amount installed depends on the selection of certain optional features. The word size is expressed in terms of two numbers: (the number of bits used for logic or control purposes) + (the number of bits used for checking the parity of the control-storage contents). When a separate control storage is provided for the service processor or channels, the table lists only the parameters of the CPU control storage.

The bus width for some models is expressed in terms of two numbers: (basic width) × (interleaving factor). The basic width is the width of the path from the storage controller to the CPU or channels. The interleaving factor indicates the number of accesses to sequential locations that can be made in one storage cycle; it applies to implementations where sequential locations are in different storage modules. The interleaving factor may be variable and may depend on the configuration. On some models the bus width is smaller than the amount of information accessed in parallel in the storage array; this is indicated by footnotes. Unless otherwise indicated, the storage-cycle time is the minimum time between successive references to the same location.

For the cache, the line-width column gives the number of bytes in the cache which are considered as one unit for addressing and replacement purposes. The first element of the product notation is the minimum transfer unit from processor storage to cache; the second element is the number of such transfer units required to make a line. Where applicable, a two-number notation is used for the cycle time to indicate the minimum time between successive read accesses and the total cache-access time. Usually, the contents of a particular virtual address in storage may be placed in only a small part of the available cache locations, where they may be found by an associative lookup. The column labeled "Associativity" shows the degree of associativity, that is, the number of different locations in the cache that may correspond to a particular virtual address.

Table 2 lists the year, month, and day when the various machines were announced and the year and month when they were first shipped.

References and notes

1. The term *architecture* is used here to describe the attributes of a system as seen by the programmer, *i.e.*, 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. G. M. Amdahl, G. A. Blaauw, and F. P. Brooks, Jr., "Architecture of the IBM System/360," *IBM J. Res. Develop.* **8**, 87–101 (1964).
3. G. A. Blaauw and F. P. Brooks, Jr., "The Structure of System/360; Part I—Outline of the Logical Structure," *IBM Syst. J.* **3**, 119–135 (1964).
4. G. M. Amdahl, "The Structure of System/360; Part III—Processing Unit Design Considerations," *IBM Syst. J.* **3**, 144–164 (1964).

5. A. Padegs, "The Structure of System/360; Part IV—Channel Design Considerations," *IBM Syst. J.* **3**, 165-180 (1964).
6. G. A. Blaauw, "The Structure of System/360; Part V—Multisystem Organization," *IBM Syst. J.* **3**, 181-195 (1964).
7. A. Padegs, "Structural Aspects of the System/360 Model 85; Part III—Extension to Floating-point Architecture," *IBM Syst. J.* **7**, 22-29 (1968).
8. D. T. Brown, R. L. Eibsen, and C. A. Thorn, "Channel and Direct Access Device Architecture," *IBM Syst. J.* **11**, 186-199 (1972).
9. R. P. Case and A. Padegs, "Architecture of the IBM System/370," *Commun. ACM* **21**, 73-96 (1978).
10. C. J. Bashe, W. Buchholz, G. V. Hawkins, J. J. Ingram, and N. Rochester, "The Architecture of IBM's Early Computers," *IBM J. Res. Develop.* **25**, 363-375 (1981, this issue).
11. W. Buchholz, Ed., *Planning a Computer System (Project Stretch)*, McGraw-Hill Book Co., Inc., New York (1962).
12. D. W. Sweeney, "An Analysis of Floating-Point Addition," *IBM Syst. J.* **4**, 31-42 (1965).
13. W. C. Carter, H. C. Montgomery, R. J. Preiss, and H. J. Reinheimer, "Design of Serviceability Features for the IBM System/360," *IBM J. Res. Develop.* **8**, 115-126 (1964).
14. C. T. Gibson, "Time-Sharing in the IBM System/360: Model 67," *AFIPS Conference Proceedings* **28** (1966 Sprint Joint Computer Conference, Boston), 61-78 (1966).
15. G. R. Blakeney, L. F. Cudney, and C. R. Eickhorn, "An Application-Oriented Multiprocessing System, Part II—Design Characteristics of the 9020 System," *IBM Syst. J.* **6**, 80-94 (1967).
16. *IBM System/370 Principles of Operation*, Order No. GA22-7000, available through IBM branch offices.
17. M. Y. Hsiao, W. C. Carter, J. W. Thomas, and W. R. Stringfellow, "Reliability, Availability, and Serviceability of IBM Computer Systems: A Quarter Century of Progress," *IBM J. Res. Develop.* **25**, 453-465 (1981, this issue).
18. *IBM 4300 Processors Principles of Operation for ECPS:VSE Mode*, Order No. GA22-7070, available through IBM branch offices.
19. H. R. Schwermer, "The ECPS:VSE Mode for the IBM 4300 Processors," *IEEE COMPCON Spring 1980 Digest of Papers*, San Francisco, Feb. 25-28, 1980.
20. T. Kilburn, D. B. G. Edwards, M. J. Lanigan, and F. H. Sumner, "One-Level Storage System," *IRE Trans. Electron. Computers* **11**, 223-235 (1962).
21. J. B. Dennis, "Segmentation and the Design of Multiprogrammed Computer Systems," *J. ACM* **12**, 589-602 (1965).
22. B. W. Arden, B. A. Galler, T. C. O'Brien, and F. H. Westervelt, "Program and Addressing Structure in a Time-Sharing Environment," *J. ACM* **13**, 1-16 (1966).
23. Only instructions published in the *Principles of Operation* are included in the counts. The following are not included: instructions available on a special-contract basis, instructions that are part of assists for specific operating systems, and instructions for emulating other architectures. For System/360, the special instructions available only on Models 20, 44, and 67 are not included. For System/370, instructions associated with the one-level-addressing facility are not included.
24. *IBM System/360 and System/370 I/O Interface: Channel to Control Unit, Original Equipment Manufacturer's Information*, Order No. GA22-6974, available through IBM branch offices.
25. P. Fagg, J. L. Brown, J. A. Hipp, D. T. Doody, J. W. Fairclough, and J. Greene, "IBM System/360 Engineering," *AFIPS Conference Proceedings* **26** (1964 Fall Joint Computer Conference, San Francisco), 205-231 (1964).
26. W. Y. Stevens, "The Structure of System/360; Part II—System Implementations," *IBM Syst. J.* **3**, 136-143 (1964).
27. M. V. Wilkes, "The Best Way to Design an Automatic Calculating Machine," *Manchester University Computer Inaugural Conference*, Manchester, England, 1951, p. 16.

Table 2 Announcement and shipment dates.

Model	*Announced*	*First shipped*
System/360		
22	71-4-7	71-6
25	68-1-3	68-10
30	64-4-7	65-6
40	64-4-7	65-4
44	65-8-16	66-9
50	64-4-7	65-8
60	64-4-7	not shipped[1]
62	64-4-7	not shipped[1]
65	65-4-22	65-11
67	65-8-16	66-5
70	64-4-7	not shipped[2]
75	65-4-22	66-1
85	68-1-30	69-12
91	64-11-17	67-10
92	64-8-17	not shipped[3]
95	*	68-2
195	69-8-20	71-3
System/370		
115	73-3-13	74-3
115-2	75-11-10	76-4
125	72-10-4	73-4
125-2	75-11-10	76-2
135	71-3-8	72-4
135-3	76-6-30	77-2
138	76-6-30	76-11
145	70-9-23	71-6
145-3	76-6-30	77-5
148	76-6-30	77-1
155	70-6-30	71-1
158	72-8-2	73-4
158-3	75-3-25	76-9
165	70-6-30	71-4
168	72-8-2	73-5
168-3	75-3-25	76-6
195	71-6-24	73-8
System/370-compatible		
3031	77-10-6	78-3
3032	77-10-6	78-3
3033	77-3-25	78-3
3033-N	79-11-1	80-1
3033-S	80-11-12	
3081	80-11-12	
4331-1	79-1-30	79-3
4331-2	80-5-7	80-8
4341-1	79-1-30	79-11
4341-2	80-9-15	

[1]Replaced by Model 65
[2]Replaced by Model 75
[3]Redesignated as Model 91
*Offered on special government contract

28. S. G. Tucker, "Microprogram Control for System/360," *IBM Syst. J.* **6**, 222-241 (1967).
29. S. S. Husson, *Microprogramming Principles and Practice*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1970.
30. P. M. Davies, "Readings in Microprogramming," *IBM Syst. J.* **11**, 16-40 (1972).
31. S. G. Tucker, "Emulation of Large Systems," *Commun. ACM* **8**, 753-761 (1965).

Table 1 Model characteristics.

Model	CPU		Control storage				Number of TLB entries	Processor storage			Cache				
	Data-flow width (bytes)	Cycle time (ns)	Size (K words)	Word size (bits)	Type	Cycle time (ns)		Size (K bytes)	Bus width (bytes)	Cycle time (ns)	Size (K bytes)	Line width (bytes)	Cycle time (ns)	Type	Associativity
System/360															
22	1	750	4	50+5	RO	750	none	24-32	1	1500[1]	none				
25	1	900	8	16+2	RW	900	none	16-48	2	900[1]	none				
30	1	750	4	50+5	RO	750	none	16-64	1	1500[1]	none				
40	2[2]	625	4	52+2	RO	625	none	32-256	2	2500[1]	none				
44	4	250	none				none	32-256	4	1000[1]	none				
50	4	500	2.75	85+3[3]	RO	500	none	128-512	4	2000[1]	none				
								1024-8192	4×(1-2)	8000[1]					
65	8	200	2.75	87+4[4]	RO	200	none	256-1024	8×2	750[1]	none				
								1024-8192	8×(1-2)	8000[1]					
67	8	200	2.75	87+4[4]	RO	200	8	256-1024	8×2	750[1]	none				
75	8	195	none				none	256-1024	8×(2-4)	750[1]	none				
								1024-8192	8×(1-2)	8000[1]					
85	8	80	2	105+3[5]	RO	80	none	512-4096	16×(2-4)	960[1]	16-32	16×4[6]	80-160	T	16
			0.5	105+3[5]	RW	80									
91	8	60	none				none	2048-6144	8×16	780[1]	none				
95	8	60	none				none	1024	8×16	180	none				
								1024-6144	8×16	780[1]					
195	8	54	none				none	1024-4096	8×(8-16)	756[1]	32	8×8	54-162	T	4
System/370															
115	1	480	20-28	20+3	RW	480	8	64-192	2	480	none				
115-2	2	480	12-20[7]	19+2	RW	480	16	64-384	2	480	none				
125	2	480	12-20	19+2	RW	480	16	96-256	2	480	none				
125-2	2	320-480[8]	16-24	19+2	RW	320	16	96-512	2	480	none				
135	2	275-1485[8]	12-24	16+2	RW	275	8	96-512	2	990[9] R 935[9] W	none				
135-3	2	275-1485[8]	64	16+2	RW	275	8	256-512	2	990[9] R 935[9] W	none				
138	2	275-1430[8]	64	16+2	RW	275	8	512-1024	2	935[9]	none				
145	4[10]	203-315[8]	8-16[11]	32+4	RW	203	8	160-2048	8	540 R 608 W	none				
145-3	4[10]	180-270[8]	32	32+4	RW	180	8	192-1984	8	405 R 540 W	none				
148	4[10]	180-270[8]	32	32+4	RW	180	8	1024-2048	8	405 R 540 W	none				
155	4	115	6	69+3	RO	115	none	256-2048	8	2070[1]	8	16	115-230	T	2
155-II	4	115	8	69+3	RO	115	128	256-2048	8	2070[1]	8	16	115-230	T	2
158	4	115	8	69+3	RW	115	128	512-6144	8	1035 R 920 W	8	16	115-230	T	2
158-3	4	115	8	69+3	RW	115	128	512-6144	8	920	16	16×2	115-230	T	4
165	8	80	2	105+3	RO	80	none	512-3072	8×4	2000[1]	8-16	8×4	80-160	T	4
			2	105+3[5]	RW	80									
165-II	8	80	4	105+3[5]	RO	80	128	512-3072	8×4	2000[1]	8-16	8×4	80-160	T	4
			1	105+3[5]	RW	80									

168	8	80	2–3.5	105+3[5]	RO	80	128	1024–8192	8×4	320	8–16	8×4	80–160	T	4–8[12]
			0.5–1	105+3[5]	RW	80									
168-3	8	80	2–3.5	105+3[5]	RO	80	128	1024–8192	8×4	320	32	8×4	80–160	T	8
			1–2	105+3[5]	RW	80									
195[13]	8	54	none				none	1024–4096	8×16	756	32	8×8	54–162	T	4
System/370-compatible															
3031	4	115	8	69+3	RW	115	128	2048–8192	8×4	345[14]	32	8×4	115–230	T	8
3032	8	80	4	105+3	RW	80	128	2048–8192	8×4	320	32	8×4	80–160	T	8
3033	8	57	3–7	105+3	RW	57	128	4096–25576	8×8	285[14]	64	8×8	57–114	T	16
			1	122+4	RW	57									
3033-N	8	57	3–7	105+3	RW	57	128	4096–16384	8×4	285	16	8×8	57–114	T	8
			1	122+4	RW	57									
3033-S	8	57	3–7	105+3	RW	57	128	4096–8192	8×4	285	0.5	8×4	57–114	T	8
			1	122+4	RW	57									
3081[15]	8	26	2[16]	104+4	RW	52[17]	128	16384–32768	8×2[18]	312[19]	32	8×16	26–52	C	4
4331-1	4	300–1600[8]	16–32[11]	32+4	RW	500[20]	64	512–1024	4	900 R 1300 W	none				
4331-2	4	200–1600[8]	32[11]	32+4	RW	500[20]	64	1024–4096	4	2600[21] R 3100[21] W	8	4×16	200	C	4
			3	32+4	RO	100									
4341-1	8	150–300[8]	14–16	32+4	RW	150	64	2048–4096	8	2400[21]	8	8×8	225	C	4
4341-2	8	120–240[8]	16–20	32+4	RW	120	64	2048–8192	16	1440[21]	16	16×4	120 R 180 W	C	8

Explanation

C Store-in-cache: On storing, the value is placed in the cache; the new value is placed in main storage at the time the cache line is reassigned or the data is requested by a channel or another processor.
K The number $2^{10} = 1024$
ns Nanoseconds
R Access for reading
RO Read-only
RW Read-write (writable)
T Store-through: On storing, the value is placed in main storage; the value is not placed in the cache unless a line has been assigned to the main-storage location.
TLB Translation-lookaside buffer, which is a part of the dynamic-address-translation mechanism
W Access for writing

Footnotes

[1] The model uses magnetic-core technology.
[2] Certain registers and paths are 17 or 18 bits wide where a main-storage address is processed in one cycle.
[3] Extended to 90 + 3 for the 1410 emulator, or 92 + 3 for the 7070 emulator.
[4] Extended to 94 + 4 when any emulator is installed.
[5] Extended to 122 + 4 for part of control storage when any emulator is installed.
[6] Although the 64-byte lines are loaded into the cache only when referred to, an entire cache sector of 1K bytes (16 lines) is assigned as a unit to a 1K-byte storage sector.
[7] The 115-2 contains a separate I/O processing unit for some functions that were executed on the CPU in a 115; hence the smaller CPU control-storage capacity.
[8] Variable, depending on the type of operation performed.
[9] Four bytes can be accessed and transferred in this time.
[10] An 8-byte-wide path is used for instruction fetch.
[11] Part of this capacity is physically in processor storage and thus has to be subtracted from the available processor-storage capacity.
[12] Depends on cache size used.
[13] The System/370 Model 195 has certain facilities (*e.g.*, time-of-day clock, control registers, MOVE LONG) not available on the System/360 Model 195.
[14] The effective transfer rate to the CPU is limited to eight bytes per CPU cycle.
[15] Each of the two CPUs has the indicated control-storage, cache, and TLB capacity.
[16] 1K of the control storage is pageable, using an area in processor storage assigned for this purpose.
[17] 26 ns when the word is available in the microinstruction buffer (*i.e.*, is within the current set of 16 words).
[18] Interleaving is on the basis of 2K bytes; no interleaving takes place within the access for a cache line of 128 bytes.
[19] An amount equal to a cache line is read or written in one storage cycle. The effective transfer rate to the CPU is limited to eight bytes per CPU cycle.
[20] 100 ns when the word is available in the microinstruction buffer.
[21] An entire cache line can be accessed and transferred between the cache and the storage unit in this time.

32. L. Bloom, M. Cohen, and S. Porter, "Considerations in the Design of a Computer with High Logic-to-Memory Speed Rates," *Proceedings of Sessions on Gigacycle Computing Systems* (presented at AIEE Winter General Meeting, New York, Jan. 29-Feb. 2, 1962), AIEE Spec. Publ. S-136, pp. 53-63.
33. D. H. Gibson, "Considerations in Block-Oriented System Design," *AFIPS Conference Proceedings* **30** (1967 Spring Joint Computer Conference, Atlantic City), 75-80 (1967).
34. J. S. Liptay, "Structural Aspects of the System/360 Model 85; Part II—The Cache," *IBM Syst. J.* **7**, 15-21 (1968).
35. C. J. Conti, D. H. Gibson, and S. H. Pitkowsky, "Structural Aspects of the System/360 Model 85; Part I—General Organization," *IBM Syst. J.* **7**, 2-14 (1968).
36. C. J. Conti, "Concepts for Buffer Storage," *IEEE Comput. Group News* **2**, 9-13 (1969).
37. K. R. Kaplan and R. O. Winder, "Cache-Based Computer Systems," *Computer* **6**, No. 3, 30-36 (1973).
38. A. J. Smith, "A Comparative Study of Set Associative Memory Mapping Algorithms and Their Use of Cache and Main Memory," *IEEE Trans. Software Eng.* **SE-4**, 121-130 (1978).
39. G. S. Rao, "Performance Analysis of Cache Memories," *J. ACM* **25**, 378-395 (1978).
40. R. A. MacKinnon, "The Changing Virtual Machine Environment: Interfaces to Real Hardware, Virtual Hardware, and Other Virtual Machines," *IBM Syst. J.* **18**, 18-46 (1979).

Received August 26, 1980; revised December 30, 1980

The author is with the IBM Data Processing Products Group located at the Poughkeepsie laboratory, Poughkeepsie, New York 12602.