IBM small-system architecture and design—Past, present, and future

by G. G. Henry

Small computer systems have become widespread and important parts of the computer industry. In this paper, a selection of IBM's small-system architectures and design approaches are reviewed. Then current architectures are discussed, with an emphasis on the RT Personal Computer. A brief presentation of trends in small systems is provided.

This paper discusses the history and current state of IBM's small computer systems with regard to architecture and design innovations and my assessment of key trends. When speaking of small systems, I will limit myself to systems with the following characteristics: they are capable of and intended for stand-alone operation (i.e., are not dependent terminals), they do not have the architecture of the System/360 or System/370, and they are relatively low-priced. Within this general group there have been many diverse IBM systems. In the last two decades, I have participated in the development of many of those systems. Not surprisingly, the set of machines that I have chosen to discuss here, in order to illustrate advances in architecture and technology and the trends in those areas, reflects my personal experience. Only selected features will be addressed, as a full description of this heterogeneous collection of systems is beyond the scope of this paper. A more extensive discussion focuses on IBM's most recent advanced small system—the IBM RT Personal Computer[1]—as the culmination of key trends and a base for future evolution.

I will attempt to support three main theses leading to my observations on future trends:

1. The unique challenges of the small-system environment have historically led to products that are innovative but are also diverse and often mutually incompatible.
2. The success of each generation of products has led to the emergence of new challenges that have further complicated the situation, but improving technology provides improved tools for the small-system designer.
3. An emerging set of common small-system characteristics, most of which are embodied in the RT Personal Computer, is increasingly evident.

For many years, the sheer power, complexity, and technological sophistication of large mainframe computer systems overshadowed the novel architectures and design features of small systems. Nevertheless, the challenges of developing competitive small computer systems are equally great and have commanded equally high levels of innovation and exploitation of advanced technology. Since the priorities that constrain small-system designers are different from those facing mainframe designers, successive generations of small systems have tended to be unique solutions to the specific situations facing their designers. In fact, in order to meet their marketplace requirements, IBM's small systems have often employed advanced architectures, design features, and technologies prior to their use in larger systems.

© **Copyright** 1986 by International Business Machines Corporation. Copying in printed form for private use is permitted without payment of royalty provided that (1) each reproduction is done without alteration and (2) the *Journal* reference and IBM copyright notice are included on the first page. The title and abstract, but no other portions, of this paper may be copied or distributed royalty free without further permission by computer-based and other information-service systems. Permission to *republish* any other portion of this paper must be obtained from the Editor.

Examples of such leading-edge features include the placement of operating system function in microcode in the System/32, the integrated relational data base of the System/38, and the data interchange architectures of the 5520 Administrative System. In many cases, small-system innovations are unique and have not yet been used elsewhere. Examples of unique small-system design features include the real-time operating system capabilities of the IBM 1800 Data Acquisition and Control System and the Series/1 and the high-level machine/single-level storage architecture of the System/38. Of course, another by-product of the responsiveness of small systems to changes in technology and architecture has been a proliferation of system families with diverse architectures and characteristics.

The pressure for innovative design solutions is further increased by new requirements arising from the recent success of the IBM Personal Computer (IBM PC) and similar systems and the intense competitive environment. Offsetting this pressure to some extent is the simultaneous rapid progress in hardware technology capabilities. However, these technologies themselves represent a challenge in that more sophisticated system design features are required to exploit them effectively.

Even with these historically diverse approaches and recent increased challenges, I perceive some clear trends in the evolution of IBM's small systems as a group. The more recent small systems, in particular, have exhibited a convergence of characteristics and a strong indication of common trends, such as "open" architecture, personal computer functions, and interactive graphic interfaces.

The most recent exemplar of these trends is the IBM RT Personal Computer, which is a confluence of key architecture and design features from previous small systems, concepts used in larger mainframe systems, and novel approaches. The RT Personal Computer includes features such as 32-bit processing, sophisticated virtual storage architecture, IBM PC compatibility features, relational data base, multiuser support, and integrated coprocessor support. Its layered, modular, and open architecture permits the addition of new software and hardware features without modification of the basic system, providing a means of exploiting subsequent hardware technology improvements in an evolutionary fashion.

In this paper, I will further develop these themes by describing (1) some of the key challenges that have historically influenced small systems, (2) some IBM small-system architecture and design innovations that have resulted from these challenges, (3) emerging challenges and implications, (4) the new IBM RT Personal Computer, and (5) the key trends and possible future directions suggested by these systems.

Historical small-system challenges and implications

All systems—large and small—are designed to have the same general objectives to be competitive and satisfy user requirements for function, performance, usability, compatibility, reliability, availability, service, and price. However, small systems generally have different priorities, trade-offs, and optimizations than larger mainframe systems. These differences are reflected in architectural and design specifics and in the technology choices for small systems.

IBM's small systems had to be usable in a single-user situation.

Historically, the pervasive objectives or challenges, some or all of which have driven small-system designs, are

- Single-user operation
- Wide range of users
- High function and performance requirements
- Cost sensitivity
- Hardware technology

These challenges are discussed in more detail in the following subsections.

Single-user operation. Although some of IBM's small systems have had multiuser capability, many have not—and they all had the requirement to be usable in a single-user situation. In a single-user installation there must be a complete set of functions adequate to install, operate, program, and maintain the total system in such a fashion that a single person, or a

small group of people, can easily understand, learn, and use the system. The direct design implications of this requirement are a consistent and usable "total system" user interface. For this reason, systems such as the System/38, 5520, and RT Personal Computer provide almost all system functions via a single consistent user interface. Included are diverse functional areas such as installation, service, file management, utility operations, and job management.

One indirect implication of this "total-system" interface is the need to integrate the underlying design structures and functions in a "single-system" architecture and design. For example, data management and communications functions are often discrete subsystems on larger systems. On such systems as the System/38 and RT Personal Computer, however, they are integrated into the base software and user interface architectures. This integration helps ensure consistent user access to these functions.

There is another major influence on systems that are primarily oriented toward a single-user environment. These systems provide a unique "console" interface to reduce the cost of a single-user configuration and to support a highly interactive interface. The user interface and underlying system functions designed for such an interface are often not practical for a multiworkstation system. The high bandwidth of the processor-to-keyboard/display of such systems as the IBM PC cannot be provided with a terminal such as a 327X. This performance is exploited by most popular personal computer programs such as spreadsheets and word processing programs. Thus, small systems have historically been divided into two fundamental design classes based on the importance of this single-user requirement: (1) those optimized for discrete terminals, such as the System/36 and System/38, and (2) those optimized for a tightly integrated console, such as the IBM PC. The RT Personal Computer merges these design points through use of innovative design and its processing power to support both types of interfaces.

Wide range of users. Designing a full-function, stand-alone system supporting a single user, regardless of that user's degree of ability and knowledge, is a difficult problem. All of IBM's small systems have had a further challenge—to support a prompted "usability" environment for users with few data processing skills while also supporting a high-function "power" environment for the more sophisticated user. It has been my experience that small systems that ignore either type of user will not be fully successful, since any potential sales opportunity area is populated by users with a wide range of knowledge, and broad acceptance and usage are required for business success. Furthermore, as individual users gain experience, they tend to migrate toward more efficient, less tutorial interfaces.

Small systems require a special focus on cost sensitivity.

The system design implication of this wide range of users is the need for multiple levels of access to functions—each providing a particular optimization relative to power versus usability. The layered system architecture of the System/38 provides this capability, whereas the RT Personal Computer provides even more layers through its multiple user interfaces, or "shells."

High function and performance requirements. These usability requirements are made more challenging by the fact that, in general, performance and function cannot be traded off for usability. This constraint is partly due to the need, discussed above, to support "power" users, but more fundamentally reflects the fact that small-system users do not want less end-user function than large-system users. Small-system users may have less data than large-system users and less extensive data sharing and communication requirements, but the requirements for application function, and thus the underlying operating system function, are similar. When this requirement for extensive function is combined with the total system interface requirement discussed above, a rich and often unique set of functions can result.

Also, performance requirements are stringent in most small-system applications—especially considering the price sensitivity considerations discussed below. The user's perception of acceptable response time does not change with the price or size of the computer. Thus, new processor architecture and op-

Table 1 Some IBM small systems

	Small Commercial	Real-Time Systems	Office Systems	Technical Professional/ Personal Computer
1961				1620
62		1720/1710		
63				
64		1800 TSX		
65				1130
66				
67		1800 MPX		
68				
69	System/3—10			
1970		System/7		
71	System/3—6		Mag Tape & Mag Card Systems	System/3—6 BASIC
72				
73	System/3—15, 4, 8, 12			
74				
75	System/32			5100
76		Series/1		
77	System/34			5110
78			3730	
79			8100	
1980	System/38		5520	
81			Displaywriter	PC
82				
83	System/36			
84				
85				
1986				RT PC

erating system designs have often been required in order to combine acceptable performance with acceptable usability and system architecture.

Cost sensitivity. Obviously, all systems are cost-sensitive, but small systems require a special focus in that regard, due in part to the extensive variety and easy availability of competitive systems in this area. Also, first-time small-system customers tend to be somewhat unsophisticated technically and are more aware of price differences than of functional variations. Competitive factors and marketing strategies of small systems often make the system *price* more directly dependent upon the hardware *cost* factors. Thus, the system designer has less freedom in what design trade-offs to consider, such as hardware support versus serviceability factors, while still meeting the price objectives of the system.

The effect of this emphasis on minimizing cost is obvious: Small systems have historically had to deal with the technical challenges described earlier within very stringent hardware cost constraints on processor power, main and disk storage capacities, I/O capabilities, etc. The interaction of cost constraints with high function and performance requirements has resulted in a series of unique system designs, each tightly tuned to a particular set of hardware capability limitations and cost relationships.

Although each system has thus satisfied *its* objectives, the total result has been a series of incompatible architectures, as hardware technology improvements have offered new system capabilities but required new system designs to exploit them.

Hardware technology. In making hardware technology choices, the designer of a low-cost system is constrained by the specified minimum entry cost and by the cost or optionality of features that affect the entire system design. For example, the system cost objectives may prohibit specifying a fixed disk as a standard design feature. This constraint, of course, greatly influences the total system architecture. More subtle effects are caused by the separability of processor functions (e.g., discrete floating-point units), the size and cost of random access memory (RAM) increments, power and heat dissipation, and physical space requirements. Historically, the hardware technology available to small systems has sig-

nificantly influenced pervasive system design characteristics. With increasing use of advanced very-large-scale integration (VLSI) technology, the impact on system design becomes less severe.

The advent of VLSI has affected the design of small systems in two distinct ways. First, it has mandated the creation of powerful design and validation tools. These tools perform the task for which they were created—simplifying and disciplining the design of VLSI chips—and also permit the chip designer to move from one chip technology to another more quickly and with less effort. Thus, new technologies can be exploited quickly and cheaply, with little change to the actual architecture of the VLSI components. The second influence of VLSI is a reduction in the fraction of the cost of a computer that is represented by the cost of its logic components. With the cost of small systems rapidly approaching an asymptote formed by the cost of their frames, covers, power supplies, and manufacturing costs, system designers' attention must be on rapidity of response to improved chip technologies and cleverness of software design.

IBM small-system highlights

As I indicated earlier, IBM's small systems have consistently had to employ new architectures, design approaches, and technologies in order to respond effectively to the general requirements that apply to all small systems. The following subsections provide a very brief description of some early IBM small systems and their unique requirements and features. I have grouped the systems into four general classes based on their characteristics and originally intended use: small commercial systems, real-time systems, office systems, and technical professional/personal computers. Table 1 shows the chronological relationship of these systems, and Table 2 summarizes the major innovations they introduced.

Small commercial systems. R. L. Taylor[2] provides an excellent summary of the line of small commercial systems developed at IBM in Rochester, Minnesota: the System/3, the System/32, the System/34, the System/36, and the System/38. A few examples here again illustrate the early application of advanced design techniques in addressing small-system requirements.

The System/32—a small commercial system, introduced in 1975 and primarily oriented toward single-user environments—placed portions of its operating system function in microcode. This arrangement provided a significant performance improvement over using the higher-level machine instruction set of the System/32. This approach has been significantly extended on the subsequent System/34, System/36, and System/38 and, of course, on larger mainframe systems.

Table 2 Major innovations in IBM's small systems

IBM System	Introduced	Innovation
1130	1965	Single-user disk operating system
		Single-user graphic design system
1800 MPX	1968	Real-time multitasking operating system
System/3	1970	Interpretive virtual storage
Model 6		Interpretive BASIC system
Basic System		Single-user workstation system
System/32	1975	Single-user commercial workstation system
		Operating system function in microcode
System/38	1978	High-level machine interface
		Single-level storage
		Integrated relational data base
5520	1980	Distributed operating system
		Office interchange architectures
		WYSIWYG text processing
PC	1981	Open hardware architecture
RT Personal Computer	1986	RISC-based processor
		Terabyte virtual memory
		Concurrent PC AT coprocessor
		Industry and IBM PC compatibility
		Additional open architecture features

Real-time systems. Harrison, Landeck, and St. Clair[3] provide a thorough review of IBM's real-time systems. These systems have been driven by some unique requirements beyond the common influences discussed above: hostile operating environment, reliability, unattended operation, sensor I/O subsystems, very flexible configurability, and, of course, short response times. The result was a series of specialized systems, each with a different architecture and software: the IBM 1720/1710, 1800, System/7, and Series/1. Within these major systems, there were even more variations, such as multiple operating systems for the 1800 and Series/1.

As an example of the creativity applied to the significant challenges of these systems, consider the 1800. In 1966, the 1800 TSX operating system was the first IBM system to provide real-time/interactive foreground support concurrent with background batch operations. The 1800 MPX operating system, introduced in 1967, was the first IBM operating system to provide multiple fixed storage partitions for scheduling a mixture of interrupt drivers and batch jobs.

The pioneering work on real-time systems validated and refined design approaches and technologies.

It also was the first IBM system to support both bisynchronous and start/stop communications and to share disk files across multiple systems. Additional "firsts" included its recovery features, such as supporting repair actions on I/O devices without having to stop the total system, automatic re-IPL (Initial Program Load) and restart, and multiple levels of automatic backup.

Although these systems seem to be a separate branch of the IBM small-system family, they have had important influences on other small systems. The early requirements in real-time systems for interrupt responsiveness and high levels of multitasking later became important requirements in all small systems. Thus, the pioneering work on real-time systems validated and refined design approaches and technologies that later received broader application. Also, many of the key designers of the later small systems had previously worked on these early real-time systems.

Office systems. F. T. May[4] describes IBM's computer-based office systems, including the recent Displaywriter and 5520 Administrative System (both introduced in 1980). As an example of their technological contributions, the 5520 implemented the first office system data stream and interchange architectures, later extended to become the IBM Revisable-Form-Content Document Content Architecture and Store-and-Forward architecture supported on most IBM systems. The 5520 had a layered, distributed operating system implemented in multiple processors of three different types. Its WYSIWYG (what-you-see-is-what-you-get) text processor—a breakthrough at the time—is now considered a fundamental requirement for small systems.

The 5520 and Displaywriter hardware and software were designed as closed systems to satisfy application requirements. Thus, they faced stiff competition from the personal computers that appeared at about the same time, which were able to do word processing almost as well while offering more flexible open architectures.

Technical professional/personal computer systems. The IBM 1620 system, introduced in 1961, represented a breakthrough in price for a small easy-to-use computing system. It was a diskless, batch-oriented, single-task system primarily aimed at FORTRAN application environments. The 1620 led to the improved IBM 1130 Computing System, introduced in 1965. The 1130 had a single-user, single-task, disk-based operating system and supported a wide range of I/O attachments. It provided one of the first integrated disk operating systems with a consistent total system user interface. Also, the 1130 with its attachment of the IBM 2250 Display Unit for graphics (the predecessor of the current IBM 5080 Graphics System) provided the first stand-alone computer-aided design system. (It also supported one of the first "arcade" games—albeit at a high price. I personally remember playing a sophisticated "space wars" game on an 1130-2250 configuration in 1967.)

The 1130 was followed by the little-known System/3 Model 6 BASIC system, introduced in 1970. Seven years before the Apple II® and 11 years before the IBM PC family were available, this system provided a disk-based operating system with an integrated, interactive BASIC language interface supporting a CRT display. Significant innovations in this system were stimulated by the fact that it did the above in 8K bytes of main storage using a processor with limited capability and performance for this application. To support this minimal storage size, an interpretive virtual memory was implemented along with sophisticated disk optimization functions such as automatically reorganizing the disk between keystrokes. Although not commercially successful, the System/3

Model 6 BASIC system established a technology base for this type of system, as well as being a training vehicle for people influential in the design of subsequent small systems. The subsequent IBM 5100 and 5110 systems, introduced in 1975 and 1977, continued the evolution of this approach.

The introduction of the IBM Personal Computer in 1981 marked the beginning of open architectures in IBM's small systems. The IBM PC was deliberately designed to allow the addition of hardware and software features by third parties. An open architecture is an acceptance of the fact that no designer can anticipate all of the uses to which his or her product will be put. The vast proliferation of hardware and software products that use the IBM PC as a base is an object lesson in the value of humility.

The most important characteristic of the IBM PC, however, is affordability. It is with the IBM PC that the value of a computer to an individual worker became high enough to justify the purchase of an entire computer *for that worker alone.* The sense of autonomy the personal computer provides its users, compared to that of terminal users, is one of the major selling points of personal computers.

Table 3 Small-system challenges and implications

Challenge	Architectural and Design Implications
Single-user operation	"Total System" User Interface leads to integrated system design User Interface optimized for high-bandwidth console
Wide range of users	*Both* a usability-oriented *and* a power-oriented interface lead to a layered system design
High function and performance requirements	Rich function Design features to provide performance within severe constraints lead to unique architectural approaches
Cost sensitivity	Unique architectures and approaches
Hardware technology	Major shifts in system design directions and limitations Limited evolutionary capability
IBM PC compatibility	Major impacts on system design and additional cost
Industry compatibility	Major impacts on system design and additional cost Potential conflicts/trade-offs with other system features
Open architecture	Primitive set of services Modular design Full interface documentation

Emerging challenges and implications

Recently, new influences from the personal computer industry and its applications have added new challenges for new small systems such as the RT Personal Computer:

- IBM PC compatibility
- Industry compatibility
- Open architecture

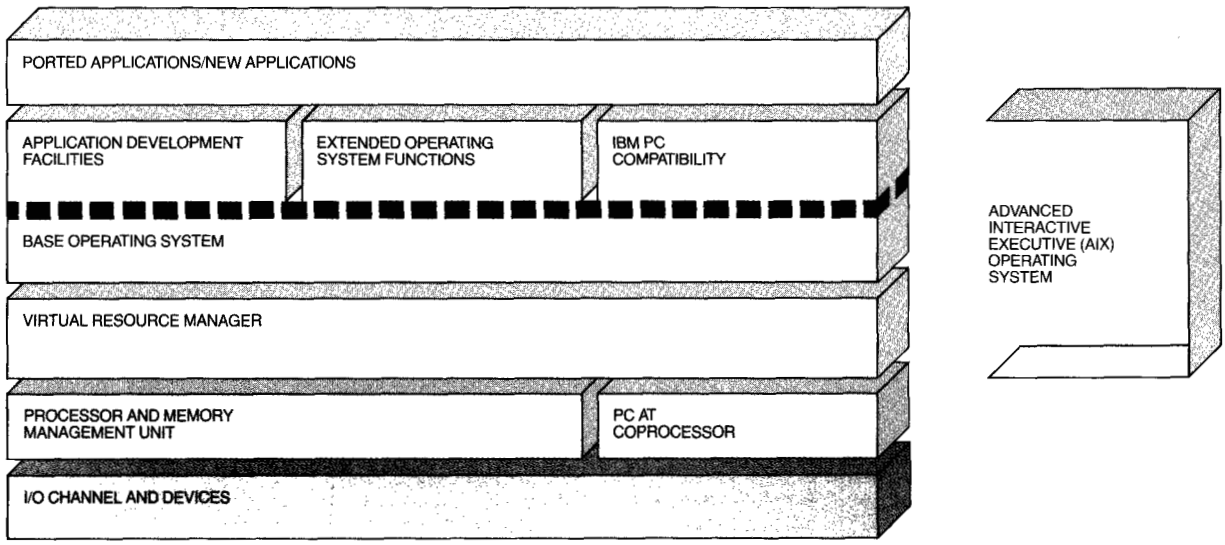
These challenges are discussed in more detail below, and the total set of small-system challenges and key implications are summarized in Table 3.

IBM PC compatibility. The ubiquitous acceptance of the IBM PC family has generated a new challenge for contemporary small systems—high levels of IBM PC compatibility. An easy design solution is to "be" a fully compatible member of the IBM PC family. However, that choice may be inappropriate given other objectives for the small system. Thus, systems such as the IBM System/36 PC and RT Personal Computer are constrained by requirements to have the appropriate architecture for their objectives, which are not those of the IBM PC, as well as providing substantial elements of IBM PC compatibility. This task is nontrivial, affecting price as well as architecture, design complexity, and usability.

Industry compatibility. A requirement similar to that of IBM PC compatibility has arisen for compatibility with non-IBM small systems and industry architectures, such as high-level language definitions, standard communications protocols, and the UNIX® operating system program execution environment. This phenomenon is driven by several factors, but primarily by the need for a system to have more software than IBM can provide. Thus, a system structure that facilitates non-IBM software development may be very important to IBM's future small systems. For example, the IBM RT Personal Computer combines a high degree of UNIX operating system compatibility with a significant amount of IBM PC compatibility, providing *both* UNIX- and PC-compatible user interface shells, file access methods, etc.

Open architecture. Related to these compatibility requirements, but with quite different implications, is the recently emerged requirement for open architecture and design. The wide variety of hardware features and application functions required by small-

Figure 1 Logical structure of the RT Personal Computer



system users means that no single manufacturer can anticipate or provide the full spectrum of configurations and software. Responsible small-system designers must, therefore, allow for the addition by the customer or by third parties of specialized hardware and/or software.

Whereas IBM PC and industry compatibility requirements can be satisfied by providing already-defined functions and interfaces, the open architecture requirement is, by definition, satisfied by no predictable, discretely implementable set of functions. Rather, it implies the need for services to permit and to help users and third parties to extend the IBM-provided hardware and software in many diverse fashions, while retaining total-system integrity. Of course, the requirement for system integrity implies interfaces with well-structured architectures, modular installation of new functions, and hardware security and integrity features.

The IBM PC family was the first IBM system to support and exhibit an open architecture. The RT Personal Computer provides significant additional support in this area, such as a resource manager component (the Virtual Resource Manager, or VRM) with full support for complex I/O services, dynamic installation of user code, full documentation of hardware interfaces, and a modular and disciplined operating system structure.

The IBM RT Personal Computer

The IBM RT Personal Computer system, announced in January 1986, builds on the techniques used in its predecessors but adds new approaches and advanced technologies. (It should be noted that the key designers of the RT Personal Computer were also key designers of the IBM System/38 and the 5520 Administrative System.) The RT Personal Computer also addresses the challenges arising from the success of the IBM PC family and industry factors such as the emergence of high-function workstations.

The major architecture, design, and technology characteristics of the RT Personal Computer are

- A two-million-instruction-per-second, "Reduced Instruction Set Computer" (RISC)-architecture, 32-bit processor
- A 40-bit virtual memory capability
- A multitasking and multiuser operating system
- Integrated IBM PC and industry system compatibility features
- An open hardware and software architecture

Although it represents a significant increase in power, capacity, and performance over IBM's previous small systems, the RT Personal Computer is distinguished from a System/370 of comparable power by the same small-system optimizations discussed earlier: single-

user orientation, a focus for users not oriented toward data processing, stand-alone environment, cost sensitivity, and IBM PC and industry affinities. Thus, the RT Personal Computer is a different type of system, with different trade-offs and optimizations from those of a mainframe system, even though it may be said to have mainframe power and sophistication.

Following is a brief description of the major design decisions and optimizations in the development of the RT Personal Computer, and the resulting hardware and software structure. Figures 1 and 2 provide a more detailed look at the RT Personal Computer structure.

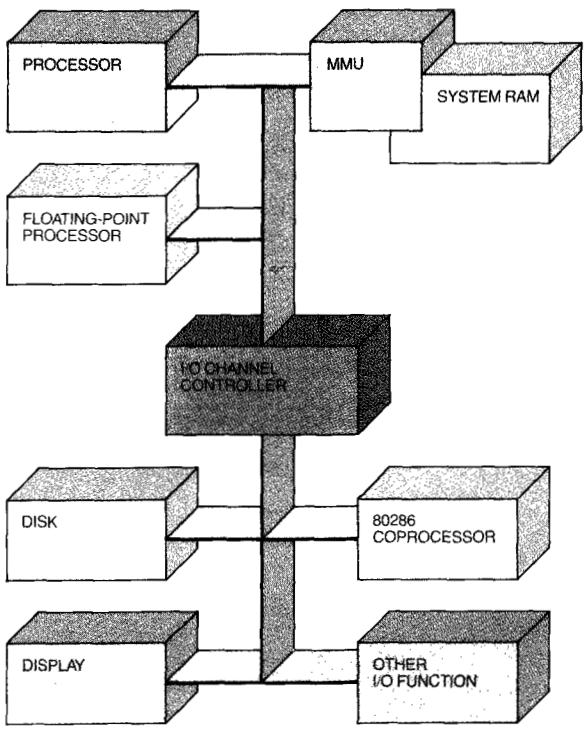
I/O channel and devices. The I/O structure chosen for the RT Personal Computer is basically the IBM PC AT 16-bit I/O channel, with some performance improvements. This choice makes it possible to use most of the existing IBM PC I/O attachment cards, while providing acceptable levels of I/O performance for native RT Personal Computer I/O devices. In addition to compatibility with the PC AT, the RT Personal Computer I/O channel provides more usable capacity to I/O devices, since processor RAM is not connected to the I/O channel and the RT Personal Computer I/O Channel Controller includes performance-assist features such as 32-bit "assembly" burst transfers and address relocation functions.

Processor and MMU. One of the most distinctive features of the RT Personal Computer is the processor and associated Memory Management Unit (MMU). In spite of the need to encourage migration of existing applications, the designers chose to use IBM's new ROMP (Research/OPD Micro-Processor).

There were a number of reasons for choosing the IBM ROMP. The ROMP provides a full 32-bit architecture, with high performance (approximately 2 million instructions per second). It has a Reduced Instruction Set Computer (RISC) architecture, making it particularly suitable as a target for compilers. Since almost all application code today is written in high-level languages, suitability as a compiler target is of much more importance than providing a large instruction set for the convenience of assembler-language programmers.

The memory management unit chip was designed to work in combination with the ROMP to provide advanced virtual storage capabilities. For example,

Figure 2 Physical structure of the RT Personal Computer



the 32-bit processor address is extended to a 40-bit virtual address with a hardware-managed inverted page table translation technique. This design significantly reduces the size of the page tables for the large virtual address space while providing very fast virtual-to-physical address translation.

Unlike the IBM Personal Computer, the RT Personal Computer carries its major system components on separate cards that plug into the system board. Thus, the ROMP and MMU are on a card that controls its own timing independently of the other components of the system. All of the memory and the optional Floating-Point Accelerator are also on separate pluggable cards.

Virtual Resource Manager. The architects of the software structure for the RT Personal Computer decided to build a Virtual Resource Manager (VRM) to control the real hardware of the machine. The VRM presents operating systems with a Virtual Machine Interface (VMI) that not only conceals the complexities of virtual memory management and

numerous I/O device types, but provides the guest operating system with a significantly more powerful set of functions than are available on the bare machine. It is therefore not accurate to think of the VRM as a pure hypervisor, like VM/370. The VMI is, in effect, a higher-level machine to which guest operating systems can be converted.

An IBM PC DOS shell on the AIX operating system allows DOS command syntax and sematics to be used to invoke AIX functions.

The VRM was necessary for several reasons. The UNIX operating system kernel that the designers wanted to run on the RT Personal Computer was not built to run on a computer with virtual memory, did not provide real-time I/O capabilities, and did not provide dynamic installation and configuration functions. The designers decided to provide these important functions "under" the kernel, rather than making extensive modifications to it. For example, the VRM provides a pre-emptive interrupt-based I/O structure, virtual storage management functions, and dynamic loading and binding of I/O device drivers. This allows such enhancements as paging of the kernel and implementation of complex, multitasking I/O device drivers.

The designers of the RT Personal Computer wanted to achieve a higher degree of program isolation from the hardware details than is possible with the current personal computer operating systems. For example, generic device classes are supported at the VMI, allowing high levels of device transparency and I/O redirection.

It was important that the optional PC AT hardware coprocessor be able to execute concurrently with ROMP without the necessity of making major changes to the existing software base and with minimal overhead for resource allocation and management.

The presence of the VRM allowed the implementation of the RT Personal Computer operating system to be isolated from changing hardware characteristics during the development process. Although this was an internal IBM requirement, the success of the VRM in meeting this goal demonstrates the success of its architecture and design features in providing high levels of hardware transparency to user software.

Operating system and extensions. As the base for the Advanced Interactive Executive (AIX™) operating system of the RT Personal Computer, the system architects chose AT&T's UNIX System V operating system. The UNIX operating system was chosen because it provides considerable functional power to the individual user, provides multiuser capabilities where needed, is open-ended, and has a large user and application base. The designers concluded that AT&T's System V version suited the RT Personal Computer better than alternative versions of the UNIX operating system because of the larger number of applications that had been built to run on that base, as well as for a variety of practical reasons.

The decision to use the UNIX operating system, however, took into account the need to make significant extensions and enhancements to meet the requirements of projected customers and target applications. This decision is, of course, the classical tradeoff between choosing an existing software system for its pragmatic characteristics and developing a new system with (hopefully) fewer deficiencies but limited applications and user familiarity. The choice was to start with the UNIX system and fix the deficiencies while retaining upward compatibility for all UNIX System V interfaces.

Some of the major enhancements made were the following:

- A usability package to provide easier access to the capabilities of the UNIX command language and to simplify the implementation of full-screen dialogues.
- Multiple, full-screen virtual terminal support to permit a single user to run several interactive applications concurrently, time-sharing the console display.
- Enhanced console support including extended American National Standards Institute (ANSI) 3.64 controls, color support, sound support, and mouse support.

- An indexed data management access method that is integrated into the base UNIX file system structure (this allows UNIX system utility functions such as "cp" to operate transparently on composite data management objects consisting of an index file and a data file).
- Extensions to exploit use of the virtual storage support; in particular, mapped file support which allows an application to map a file into a 256-megabyte virtual address space and access it with loads and stores rather than reads and writes. A derivative is used by the system to provide mapped text segment support, allowing demand paging from the program libraries.
- Enhanced signals to allow flexible exception-condition handling.
- A variety of floating-point support functions.
- Simplified installation and configuration processes.

Loucks[5] gives a much more extensive and detailed description of the structure of the AIX operating system.

IBM PC compatibility. The objective of easing user and application migration required not only UNIX application portability but also a high level of program and user interface compatibility with the IBM PC family. The hardware requirement was satisfied by an IBM PC AT hardware coprocessor that includes an Intel 80286 processor along with associated hardware to provide a high level of PC AT hardware compatibility.

In addition to the hardware compatibility provided by the coprocessor, an IBM PC DOS shell on the AIX operating system allows DOS command syntax and semantics to be used to invoke AIX functions, and IBM PC compatibility modes in the RT Personal Computer BASIC and Pascal compilers provide IBM PC BASIC- and Pascal-compatible functions.

The AIX shell also includes IBM PC diskette access utilities and access methods to facilitate data migration and portability.

One of the key technical decisions relative to compatibility was to allow the PC AT coprocessor to execute IBM PC programs concurrently with ROMP programs, sharing system resources such as main storage and the system console. This unique capability is provided by a combination of the coprocessor hardware card and the VRM. The VRM manages the allocation and sharing of resources in such a way that the concurrent execution of the coprocessor is transparent to the operating system and application programs. For example, the VRM allocates the console keyboard to either the coprocessor or the ROMP and monitors keystrokes for a "hot key" sequence signaling a need to switch to the other processor. In a similar fashion, other system resources are managed so that the coprocessor applications seem to execute in a virtual terminal just as the ROMP applications do.

Small-system trends

In the history of the development of IBM's small systems, there have been three general kinds of trends: those driven by the general advance of hardware and software technology, those compelled by improved understanding of the challenges and the advantages and disadvantages of different approaches, and, most recently, those motivated by a changing user audience and requirements. As mentioned previously, there has also been a pervasive influence of the cross-pollination of key design elements and personnel across systems.

Subtler architectural trends may be more important in the long run.

Among technology-driven trends, I include the obvious "bigger (in capacity), smaller (physically), faster, cheaper" changes. The very rapid growth of the IBM PC capabilities driven by the apparently insatiable user demands—explicit or implicit—for more function, capacity, and performance is the obvious manifestation of this trend. The more important factor is the change in system architecture that permits accelerated adoption of new technological advances. Although these trends presumably cannot continue forever at their current geometric rates, they will undoubtedly carry small-system capabilities further than anyone can now foresee.

The subtler architectural trends are more difficult to identify and describe but may be more important in the long run. At the moment, I see three major areas that will occupy system designers' attention in the foreseeable future:

- Distributed architectures
- Noncoded data (image and sound)
- Improved user interaction

I mentioned earlier the importance of autonomy to the personal computer user. As the amounts of storage available to personal computer users grow, however, the existence of substantial amounts of data that are not controlled by or available to the organization in general becomes a significant concern. The data base of an organization is one of its most significant assets—in many cases ranking just after the skills, knowledge, and commitment of its people and above its physical plant. The existence of increasing amounts of parochially held data is a threat to the efficiency of the organization.

To ameliorate this situation, it will be necessary to devise ways of making the growing body of data held on small systems available to the remainder of the organization. In some cases, a solution will involve local-area networks, whereas in other cases it will include mainframes as data repositories and interrogators. As with the need for open hardware and software architectures in general, it is clear that there will be a future requirement for a high degree of architectural eclecticism in communications and data access structures to allow the necessary distributed systems to be built.

The increasing processing speeds and information storage capacities of small systems are also making it practical to store large quantities of noncoded image and audio data. As with the other aspects of small-system architecture, noncoded information structures will have to prepare for the unexpected. Designers will have to provide ways for developers to use image and audio data without constraining the resulting applications.

By this time, the knowledgeable reader will probably have asked, "What about artificial intelligence?" I believe that one of the most significant and complex challenges for system designers in the coming decade will be to find ways of making increasingly powerful and capacious small systems into flexible, responsive, and usable tools for a wide spectrum of users. Some of the techniques will come from the research into expert systems, some from the psychological investigation of how users comprehend and use computer systems, and some from psycholinguistic research into the differences between the way people understand and use words and the way computers process words. If small systems are to be of use in containing and managing the "information explosion," designers of future hardware and software will have to take into account not only the capabilities and limitations of computers, but also the strengths, weaknesses, and individual differences of the human beings with whom those computers will interact with increasing intimacy.

Summary

In the last quarter-century, IBM's small systems have advanced from being fairly simple systems, severely constrained by their cost objectives, to a level of sophistication equal to and in some cases exceeding that of mainframe systems. This phenomenon has been driven by a simultaneous improvement in the technologies available to small-system designers and intensification of the challenges that they must address. The last five years especially have seen rapid advances in the technology as well as significant new design challenges.

The latest IBM small system—the RT Personal Computer—represents a major step in small-system design in terms of its use of advanced hardware technology and its architectural base addressing a wide range of requirements. However, clear trends relative to technology, user requirements, and continuing application growth will continue to drive small-system designs to higher levels of sophistication, hopefully in evolutionary ways.

I look forward to the developments of the coming years, in which I expect the balance of innovation and marketplace growth to shift even more toward small systems, as user requirements and competitive pressures continue to grow and advances in hardware technology enable continued advances in small-system capabilities.

Acknowledgments

The numerous advances of IBM's small systems over the years have, of course, allowed this paper to be written. My congratulations and respect go to all of the many people who have contributed to this exciting aspect of IBM's success. My specific thanks go to Frank Waters of IBM Austin, who made this paper

happen. He did everything from typing, to editing, to goading me into action, and most important, he made many significant technical suggestions.

Apple II is a registered trademark of Apple Computer, Inc. UNIX is a registered trademark of AT&T Bell Laboratories.

Cited references and note

1. RT, RT Personal Computer, and RT PC are trademarks of International Business Machines Corporation.
2. R. L. Taylor, "Low-end general-purpose systems," *IBM Journal of Research and Development* **25,** No. 5, 429–440 (September 1981).
3. Thomas J. Harrison, Bruce W. Landeck, and Hal K. St. Clair, "Evolution of small real-time IBM computer systems," *IBM Journal of Research and Development* **25,** No. 5, 441–451 (September 1981).
4. F. T. May, "IBM word processing developments," *IBM Journal of Research and Development* **25,** No. 5, 741–753 (September 1981).
5. Larry Loucks, "IBM RT PC AIX kernel—Modifications and extensions," *IBM RT Personal Computer Technology*, IBM Corporation, SA23-1057 (1986), pp. 96–109; available through IBM branch offices.

G. Glenn Henry *IBM Industry Systems Products, 11400 Burnet Road, Austin, Texas 78758.* Mr. Henry is an IBM Fellow, and was the manager of hardware and software system development for the IBM RT Personal Computer. He joined IBM in 1967 in San Jose, California, and has been involved in the design and management of the IBM 1800, System/3, System/32, and System/38. He has received several formal awards, including an IBM corporate award in 1982 for his work on the System/38. Mr. Henry received a B.S. and an M.S. in mathematics in 1966 and 1967, respectively, from the California State University at Hayward. He is a member of the ACM and IEEE.

Reprint Order No. G321-5278.