The current state of minicomputer software

by JOSEPH F. OSSANNA
Bell Telephone Laboratories, Inc.
Murray Hill, New Jersey

INTRODUCTION

No one who has watched the spectacular growth of the minicomputer industry needs to be told that people are finding new small computer applications at an astonishing rate. Undoubtedly there are many reasons for this striking increase in minicomputer utilization. Certainly one reason is the increasing sophistication of minicomputer software. This paper will summarize the development of this sophistication, and will try to characterize its present state. Minicomputer hardware evolution will be discussed because of its effect on minicomputer software evolution. The range of minicomputer uses and users will be characterized along with their varying software organizations. Commentary on techniques of minicomputer software development is included. A number of examples of both past and present minicomputer software systems are presented.

MINICOMPUTER HARDWARE

A minicomputer is a "minicomputer" by nature of its hardware which is relatively simple, physically small, and comparatively inexpensive. This is accounted for by the following general characteristics compared to the "ordinary" large computer.

- Shorter wordlength
- Smaller memories
- Fewer and simpler instructions
- Simpler input/output capabilities
- Minimal features
- Fewer peripherals

In addition to the above, limited software support by the manufacturer and lower marketing costs have contributed to the lower price of the minicomputer. Wordlengths range from 8 to 24 bits with 16 bits being the most common in currently available machines. The range of memory sizes available in a typical current machine is from 4K words (K = 1024) to 32K words. Basic instruction sets typically lack multiply and divide, decimal and floating-point arithmetic, arbitrary shifting, and fancy character string manipulation. Although minicomputer memories are smaller than those of large machines, the memory cycle times frequently are similar, with the result that computing speeds are similar in applications where a smaller word size and the absence of some types of arithmetic are not a factor. Basic input/output structures usually employ a single time-shared bus with limited device addressing and with limited parallelism during data transfer. Features such as memory protection, relocation, and segmentation are usually nonstandard or unavailable. Often few fast registers are included. The selection of peripherals standardly interfaced to a given machine typically has been limited. In the past, the only reliable peripherals available have been expensive ones originally designed for the larger computer systems.

To some extent, what has just been said above is becoming less true with time. The spectacular advances in integrated circuit technology in recent years have resulted in dramatic price declines. Core memory and peripherals have also become less expensive. The price of a basic system consisting of the central processor (cpu), 4K×16 bits of core memory, and a Model 33ASR teletypewriter has dropped from about $25,000 in 1966 to about $8,000 in 1971.1 The circuit technology advances have also made increased system capability economically attractive. The following amendments to the list of minicomputer characteristics are presently apropos.

- Instruction sets improving; fast multiply, divide, and floating-point optionally available. More and better addressing modes being designed.
- Memory protection, relocation, and segmentation becoming optionally available.
• Larger sets of fast general registers becoming common.
• New bus concepts are providing better input/output structures.
• Priority interrupt structures and interrupt vectors are becoming standard.
• Smaller and well designed peripherals are available at reasonable cost.
• Less expensive disk storage is becoming available in sizes commensurate with small machines.

Manufacturers have achieved this and are providing increased software support without noticeably stemming the price decline.

These recent improvements in small machine architecture have attracted more sophisticated applications and more sophisticated systems designers. The advent of inexpensive disk storage is probably the most important occurrence influencing minicomputer software evolution.

A tabular summary of currently available minicomputers and their characteristics can be found in Reference 2. Similar information along with projections of the future of the minicomputer industry are given in Reference 1.

MINICOMPUTER USES AND USERS

Most minicomputer systems consist of a general-purpose small machine configured in a way suitable for some particular application. Tailoring the size and complexity of a system to the requirements of a particular job can result in an economy of specialization that outweighs the economy of scale of using a part of a larger machine. Accordingly, one would expect the range of applications to break down in a way that reflects this economic bias. Auerbach reports that in 1970 minicomputer dollar sales were divided as follows.

45% Industrial and process control.
3% Peripheral control (COM, OCR, line printers, etc.).
20% Communications control (concentrators, switching).
10% Computations.
22% Data acquisition and reduction.

Process control and data acquisition often involve intimate connection to distributed sensors or laboratory instruments. This is often infeasible for centrally located large machines, especially when coupled with the necessity for rapid control by the user and his program.

In a survey conducted within Bell Laboratories in the Spring of 1971 of approximately 200 minicomputer systems the following usage breakdown was reported.

31% Computerized testing of equipment and devices.
19% Data acquisition and reduction.
18% General-purpose computing, some with data acquisition.
15% Peripheral control (plotters, cameras, displays, unit-record equipment).
9% Intelligent graphic terminals and interactive graphic systems; general-purpose graphic-oriented computing.
7% Process control.

It is possible to observe different software organizations across these various types of users. Some of the differences are clearly due to the nature of the applications. Many are due to the nature of available software support, to the software experience of the programmers involved, and to hardware limitations. In the next section these differences are characterized.

SOFTWARE ASPECTS

The choice of a software organization for a minicomputer system can profoundly affect hardware requirements, general efficiency, the range of applications, extensibility, and the ease of software maintenance and modification. The best organization in a particular case depends on which considerations are paramount. Other considerations include the following.

• Are the application programs to be run under an operating system?
• Is there to be on-line secondary storage?
• Are there crucial real-time response requirements?
• Are there to be multiple simultaneous users?
• Is the system to be self-supporting—i.e., can the application programs be developed and maintained on the system?
• Is there a justification or need for core protection, relocation, or some sort of virtual memory?

An "operating system" is a collection of basic and frequently used utility programs essential for convenient and efficient use of a machine. For many people this loose definition means the sum of all the programs they use that were written by someone else and consequently not easily modified. Most minicomputer operating systems are simple and basic compared to those used...
Such inclusion of operating system functions in a user's program, other support programs are needed. Implications for software system design. By real-time minimization of core usage is paramount, the generality is usually the foundation of the better general-purpose system. An assembler or compiler is needed to translate this source text into executable machine code. Optionally a debugger is useful for analyzing program troubles. Operationally, the first two are run like user programs, and the latter is usually combined with the user program if needed.

If there is justification for on-line random-access secondary storage such as a disk or drum, a different perspective is possible. The less frequently used portions of any operating system and of the user's program need not be permanently core resident, and core requirements can be minimized without foregoing the convenience and generality of an operating system. Further, support programs like those mentioned above can be stored on disk for handy use (by disk will usually be meant any such secondary device). Additional support in the form of libraries of user and utility programs can be easily kept along with a binder program that can search libraries and combine collections of program modules into executable units. Greater convenience is obtained when the operating system provides organized access to the disk by means of some sort of a file system. A well designed file system is usually the foundation of the better general-purpose operating systems.

Real-time response requirements can have strong implications for software system design. By real-time we do not here mean the requirements of controlling conventional peripherals such as disks, tapes, unit-record equipment, or most communication lines. In these cases a too-slow response usually causes retries and degrades efficiency but is not ordinarily fatal. What is meant are the response requirements of controlling such things as a steel rolling mill, a linear accelerator, or an oil refinery. Industrial process control systems typically receive a variety of external stimuli requiring a wide range of response times. This necessitates a hierarchical priority structure for the interruption and execution of programs. When there is more than a few such programs the scheduling can become quite complex. Operating systems for such situations are usually different than general-purpose ones; process control systems are discussed later. Data acquisition systems and communications control systems can have real-time response problems, if peaks can occur that would temporarily tax their processing capacity and it is important that no data be lost.

Unlike a process control system which supervises known external processes with generally known characteristics, a multi-user system such as a general-purpose time-sharing system faces user submitted tasks of more or less unknown characteristics. For example, resources such as processor time, core space, and file system space demanded by the user processes vary dynamically, and the system generally must schedule adaptively and enforce various limits on resource utilization. Even if a minicomputer system is economically justified when wholly committed to some application, a multi-user capability can permit software development to take place while the main application is running. An example of a multi-user system is given later.

Core protection is a hardware feature that enables the operating system to make sensitive regions of core storage inaccessible, thereby protecting those regions from damage by being written or preventing information from being read. This feature is uncommon in minicomputer systems that operate with debugged programs, but is usually mandatory in multi-user systems that permit users to write and execute arbitrary programs. Relocation is a hardware feature that permits a program to be loaded at any convenient place in core memory, and to be moved (while interrupted) to another place. This feature can be a great asset to, for example, a process control system trying to juggle a myriad of different sized programs between disk and core. Virtual memory is a concept where a program can think it has more core than it really has or even than there is.

RETROSPECTION

When one looks at the minicomputer systems being used in the early 1960s one sees minimal hardware with typically no disk storage and minimal software support in the form of operating systems, compilers, editors, etc. The temptation was therefore great to make use of the nearest computer center with its big machine, big disk, and varied software support.

One of the first minicomputer applications that the
Another example is a small laboratory computer system used for the acquisition and reduction of data from a spectrometer. It originally consisted of a DEC PDP8 with 4K×12 bits of core, a Model 33ASR teletypewriter, a DEC Model 34D display, a home-made analog-to-digital converter, an extended arithmetic element, and interfaces to laboratory equipment. The user wrote his own application software in a few months using a DEC supplied editor and assembler on the PDP8 itself. Both user and support programs were stored on paper tape and had be be loaded using the reader on the 33ASR. How long does it take to read 4K×12 bits at the rate of 10 8-bit characters per second? This user eventually got 128K×12 bit drum primarily to facilitate the acquisition of more data than would fit into core. He now uses a DEC supplied disk-based operating system for program development and to run his application. This system is typical of quite a few minicomputer systems within Bell Laboratories.

A more recent (1968) example of a laboratory computer system designed for self-service by a variety of users at Bell Laboratories is described in Reference 8. The hardware consists of a Honeywell DDP224 with floating-point instructions, 32K×24 bits of core, two disk drives for 2.4 million word removable disk packs, magnetic tape, a card reader, analog-to-digital and digital-to-analog converters primarily for speech and music, a console typewriter, various graphical input and output devices, and miscellaneous devices such as a piano keyboard. Extensive software support is provided for the user in the form of a simple disk operating system with utilities for all the various devices, an assembler, a Fortran compiler, an editor, a loader, and a large library of useful programs. A simple file system is provided along with disk utilities for physical disk accessing. Disk conflicts are avoided by informal agreements and by many users having their own disk packs. The operating system will bring whatever program the user wants into core from disk but the user must manually transfer control to its starting location. This machine has been very successfully used by more than a dozen users for various research projects. A main drawback is the necessity for users to sign up for time on the machine, and a limited multi-user capability would undoubtedly be welcome. The system is much like a user-operated batch processing system. Even though some of the applications involve real-time response and are demanding of the processor, such tasks as file editing and compiling could go on at a lower priority or be done concurrently with less demanding tasks. Such an ability would require a better file system for user isolation and some scheme for rotating user programs through or sharing core. In this

Another example is a small laboratory computer system used for the acquisition and reduction of data from a spectrometer. It originally consisted of a DEC PDP8 with 4K×12 bits of core, a Model 33ASR teletypewriter, a DEC Model 34D display, a home-made analog-to-digital converter, an extended arithmetic element, and interfaces to laboratory equipment. The user wrote his own application software in a few months using a DEC supplied editor and assembler on the PDP8 itself. Both user and support programs were stored on paper tape and had be be loaded using the reader on the 33ASR. How long does it take to read 4K×12 bits at the rate of 10 8-bit characters per second? This user eventually got 128K×12 bit drum primarily to facilitate the acquisition of more data than would fit into core. He now uses a DEC supplied disk-based operating system for program development and to run his application. This system is typical of quite a few minicomputer systems within Bell Laboratories.

A more recent (1968) example of a laboratory computer system designed for self-service by a variety of users at Bell Laboratories is described in Reference 8. The hardware consists of a Honeywell DDP224 with floating-point instructions, 32K×24 bits of core, two disk drives for 2.4 million word removable disk packs, magnetic tape, a card reader, analog-to-digital and digital-to-analog converters primarily for speech and music, a console typewriter, various graphical input and output devices, and miscellaneous devices such as a piano keyboard. Extensive software support is provided for the user in the form of a simple disk operating system with utilities for all the various devices, an assembler, a Fortran compiler, an editor, a loader, and a large library of useful programs. A simple file system is provided along with disk utilities for physical disk accessing. Disk conflicts are avoided by informal agreements and by many users having their own disk packs. The operating system will bring whatever program the user wants into core from disk but the user must manually transfer control to its starting location. This machine has been very successfully used by more than a dozen users for various research projects. A main drawback is the necessity for users to sign up for time on the machine, and a limited multi-user capability would undoubtedly be welcome. The system is much like a user-operated batch processing system. Even though some of the applications involve real-time response and are demanding of the processor, such tasks as file editing and compiling could go on at a lower priority or be done concurrently with less demanding tasks. Such an ability would require a better file system for user isolation and some scheme for rotating user programs through or sharing core. In this

Another example is a small laboratory computer system used for the acquisition and reduction of data from a spectrometer. It originally consisted of a DEC PDP8 with 4K×12 bits of core, a Model 33ASR teletypewriter, a DEC Model 34D display, a home-made analog-to-digital converter, an extended arithmetic element, and interfaces to laboratory equipment. The user wrote his own application software in a few months using a DEC supplied editor and assembler on the PDP8 itself. Both user and support programs were stored on paper tape and had be be loaded using the reader on the 33ASR. How long does it take to read 4K×12 bits at the rate of 10 8-bit characters per second? This user eventually got 128K×12 bit drum primarily to facilitate the acquisition of more data than would fit into core. He now uses a DEC supplied disk-based operating system for program development and to run his application. This system is typical of quite a few minicomputer systems within Bell Laboratories.

A more recent (1968) example of a laboratory computer system designed for self-service by a variety of users at Bell Laboratories is described in Reference 8. The hardware consists of a Honeywell DDP224 with floating-point instructions, 32K×24 bits of core, two disk drives for 2.4 million word removable disk packs, magnetic tape, a card reader, analog-to-digital and digital-to-analog converters primarily for speech and music, a console typewriter, various graphical input and output devices, and miscellaneous devices such as a piano keyboard. Extensive software support is provided for the user in the form of a simple disk operating system with utilities for all the various devices, an assembler, a Fortran compiler, an editor, a loader, and a large library of useful programs. A simple file system is provided along with disk utilities for physical disk accessing. Disk conflicts are avoided by informal agreements and by many users having their own disk packs. The operating system will bring whatever program the user wants into core from disk but the user must manually transfer control to its starting location. This machine has been very successfully used by more than a dozen users for various research projects. A main drawback is the necessity for users to sign up for time on the machine, and a limited multi-user capability would undoubtedly be welcome. The system is much like a user-operated batch processing system. Even though some of the applications involve real-time response and are demanding of the processor, such tasks as file editing and compiling could go on at a lower priority or be done concurrently with less demanding tasks. Such an ability would require a better file system for user isolation and some scheme for rotating user programs through or sharing core. In this
systems. Most systems involve two forms of control: direct digital control in which the computer directly sets the operating points of controls process variables; and supervisory control that slowly modifies the control function that slowly modifies the control algorithm. On top of this response time hierarchy can be management information and long-term scheduling functions involving delays measured in days or weeks. Direct digital control tasks are usually many in number but relatively simple while supervisory control tasks are fewer in number but relatively complex.

In the next section an attempt will be made to characterize minicomputer software currently in common use and to describe some forefront efforts.

CURRENT EXAMPLES

Industrial process control

All indications are that this field will be one of the largest users of minicomputers. Work in this field, not all involving minicomputers, is relatively extensively documented. As indicated previously, process control applications often involve a wide range of real-time response requirements. Control algorithms involve the operation of complex multi-loop negative feedback systems. Most systems involve two forms of control: direct digital control in which the computer directly controls process variables; and supervisory control in which the computer sets the operating points of otherwise independent analog controllers. The response required in direct digital control ranges from milliseconds to seconds, while that required for supervisory control ranges from seconds to minutes. In addition, there is usually an overall optimizing and adaptive control function that slowly modifies the control algorithms. On top of this response time hierarchy can be management information and long-term scheduling functions involving delays measured in days or weeks. Direct digital control tasks are usually many in number but relatively simple while supervisory control tasks are fewer in number but relatively complex.

The number of task-oriented programs is usually sufficiently large that economy dictates that they not all be core resident; thus programs need to be brought into core when needed. The existence of the shorter real-time requirements as well as the desire for reasonable efficiency dictates that several programs reside simultaneously in core. Thus a process control system takes on the appearance of a multi-programmed time-sharing system. In contrast with time-sharing systems, the scheduling of the execution and interruption of programs is more complex in process control systems. The relative priority of programs is typically dependent on process conditions, operator intervention, and even such things as the time of day. While the general technique is to respond to hardware interrupts and place programs into execution priority queues, provision must usually be made for certain high priority programs to purge certain other low priority programs from queues and for low priority programs to inhibit their interruption at certain critical times. Some sort of organized provision is required for the interlocking of shared data bases. Although most of the programming is done in assembly language and in Fortran, a number of process control oriented languages have been developed and used. The process control field has in some ways reached a greater level of sophistication than other fields of application.

Communications control

This field includes communication line concentrators and controllers used as communication front-ends for large computer systems, as nodes in communication networks, and as message switching centers. The latter application requires secondary storage while the others do not. Data line concentration and control can usually be done less expensively and more flexibly with a minicomputer than with wired hardware controllers.

An example of a minicomputer front-end is the line concentrator used to interface varied terminal equipment to the Michigan Time-Sharing System. This system was experimental and its hardware complement is not necessarily optimized. It consists of a DEC PDP8 with 12K×12 bit of core memory, a Model 33ASR teletypewriter, a high-speed paper tape reader/punch (to facilitate software development), an extended arithmetic element, an interface to the local IBM 360/67's multiplexor channel, and interfaces to standard telephone data sets operating at all the standard speeds. The lower speed interfaces are bit buffered and the faster ones are character buffered. The system performs a limited form of terminal type
identification of low speed asynchronous typewriter terminals and handles a variety of IBM and teletypewriter terminals. Because the speed of operation of each line is program controlled, statistical sharing of the access lines reduces the required number of telephone ports. The system-terminal interface is better human engineered than that obtained using standard IBM hardware line controllers. Full duplex operation of teletypewriters is standard and read-ahead of the terminal keyboard is provided.\textsuperscript{13}

While such a system is less complex than many process control systems, the bit and/or character completion interrupt rate is high and the responding programs can't dawdle. The low number of distinct tasks combined with the high task rate dictates that the software all be core resident. The system described above was equipped to serve 16 low speed asynchronous lines and 4 voice grade synchronous lines; however, a system of this kind more typically can serve between 32 and 64 low speed lines along with several voice grade lines.

\textit{Paper tape operating systems}

The above heading may overdignify the collection of software that it refers to, but this kind of software support is one the oldest and is still common. If one has a typical basic minicomputer configuration consisting of a processor, 4K words of core, and a teletypewriter with paper tape handling, one either does his software development using support programs stored on paper tape or uses some other machine for development and reserves the basic machine to run the application. The main drawback with paper tape based support is the slow rate of reading and punching paper tape. This drawback can be largely mitigated by adding a high speed reader/punch. Surprisingly enough, the convenience of working on your own machine and the fact that a cycle of editing, assembling, and reloading may be shorter than the time required to travel to and wait on the local computer center, may make the use of a paper tape operating system feasible. Of course, access to a good time-sharing system with the right kind of program support or to another machine like yours but better equipped is probably preferable.

An example of a current paper tape system in DEC's paper tape software for the PDP11.\textsuperscript{14} It provides an absolute loader for reading in all programs, a reasonable text editor, an assembler, floating-point simulation, a math package, a collection of input/output utilities, and an octal debugger. The editor is used to create and edit programs at the teletypewriter; during editing, the old copy of the source is read off paper tape a block at a time, edited, and then punched. Assembling a program consists of reading in the assembler, feeding the source program to the assembler twice, and receiving an executable object copy of your program from the punch. Your program can then be tried by loading it; after loading the loader will transfer control to it automatically. If it works, fine; if not the octal debugger can be loaded into vacant core (if any) and used to poke around in the remnants. All of this is or is not as awkward as it seems depending on the alternatives open to you. With enough core a simple core resident operating system containing all the support programs with some core left for the user would be even more convenient; the additional cost is comparable to the cost of the smaller disks.

\textit{Disk operating systems}

It has been observed several times previously that the addition of disk storage to a minicomputer system makes a world of difference. Even the simplest disk-based operating system greatly magnifies the convenience of working on your own machine, which becomes like using a one-user time-sharing system. Multi-user systems are possible and desirable and are discussed later. Much of what is possible with disk is possible in a slower more restricted way using randomly accessible magnetic tapes such as DEC's DECTape,\textsuperscript{15} or even small tape cartridges.

An example of a recently available (circa 1968) disk operating system for a machine that has been around in some form for awhile is the PS/8 operating system for the PDP8 family of machines.\textsuperscript{16} It requires 8K words of core and a disk or DECTape.

Another example is DEC's disk operating system for the PDP11.\textsuperscript{17} It requires at least 8K words of core, a small disk augmented by DECTape, or a larger disk, a high speed paper tape reader/punch, and a console typewriter. The support software includes a command interpreter, a reasonable text editor, an assembler, a Fortran compiler, an octal debugger, a file utility package, floating-point simulation, a math package, and a linker to combine programs. The operating system provides a reasonable file system and does input/output for the user. Core is divided into five regions: a small region at the bottom containing the interrupt vectors which indicate the location of interrupt handling programs; next a small area for the resident monitor; an area for the user program at the top of core; below that the stack (where parameters are stored temporarily during the transfer of control between programs); and
a free area divided into blocks and used for buffers, temporary tables, and device drivers brought into core. To invoke a system support program such as the editor or a user program written to run under the system, the user types a request on the console keyboard which can include information about which input and output devices are to be used during the run. User programs can be used interactively and can be interrupted for debugging.

Multi-user time-sharing systems

It is an accepted fact nowadays that the time-shared use of a computer is one of the more effective ways of increasing the productivity of a group of users in need of computing service. The advantages of "working on your own machine" are effectively extended to the whole group at once. A multi-user system is inherently more complex than a single user one; it is necessary to multiprogram and/or swap user programs between disk and core and to keep track of everyone's input/output and machine conditions. It is helpful too, if users can be prevented from injuring each other and the system.

We shall present here, as an example of a multiuser system, a system recently developed (primarily by K. L. Thompson) at Bell Laboratories which is felt to be at the forefront of minicomputer software design. This system utilizes a DEC PDP11/20 computer with 12K × 16 bits of core storage, a 60 cycle clock, a 256K work fixed-head disk, a disk drive for 1.2 million word disk cartridges, a dual DECTape drive, 8 low speed asynchronous line interfaces for remote typewriters, and a little-used paper tape reader/punch.

Core is divided into an 8K word/region containing the totally core resident operating system and a 4K word region for the user program and his stack. The system space includes about 2K used for a pool of dynamically allocated buffers. The operating system maintains an excellent file system, creates and manages user processes executing core images), does input/output for the user, and performs all scheduling functions. Scheduling is by placing ready-to-run users on one of three queues, depending on whether the user has interacted (has just typed an action character on his console), some pending input/output has completed, or the user has run out of time allotted for execution. All users are swapped between the fixed-head disk and the same region of core.

The main feature of this system is a versatile, convenient file system with complete integration between disk files and all input/output devices. There are three kinds of files; ordinary disk files, directories, and special files representing physical devices. A directory is like an ordinary file except that only the system can modify it. The file system is a tree structured hierarchy originating at a root directory. Any type of file can occur at any level. Files have names of eight of fewer characters. At any given time a user process is associated with a particular current directory. When the user program wishes to open or create a file, it gives the system the file name of a file in the current directory or it gives a path name which either specifies the absolute location in the tree or the location relative to the current directory. If the reference is successful, the system returns a file identification number to be used for future references. File system protection consists of associating with the file at time of creation the name of the creator and permitting him to specify whether he and others can read and write the file. Files are formatless and are regarded as strings of 8-bit bytes.

Another feature of the system is the ability to initiate asynchronously executing processes from within a program or from command level. Commands are normally terminated with semicolons or new-lines; the command interpreter creates a process to execute the command and waits for its completion. If the command is terminated instead by an ampersand, the same thing occurs except that the command interpreter doesn't wait. Argument strings typed with a command are made available to the program by placing them on the stack.

The name of any executable program can be used as a command. The name is first searched for in the current directory and if that fails it is then searched for in a system library. A myriad of commands are available including those for listing directories, moving, copying, and deleting files, changing the owner and mode of a file, and changing the current directory. Other programs available include a context text editor, an assembler, a symbolic debugger, a linking loader, a dialect of Basic, Fortran, a text formatter, games such as chess, and various experimental languages.

The system operates 24 hours a day, seven days a week. A user community of about a dozen people access the machine regularly although it is not a service offering. The group that owns the machine uses it heavily for preparing documentation using the editor and text formatter. The maximum of eight simultaneous users (limited by the number of lines) does not tax the machine's capacity. Because of the lack of core protection, any user can write, assemble, and run a program that could damage the system and cause a system crash. However, since most users use debugged pro-

From the collection of the Computer History Museum (www.computerhistory.org)
grams like the editor or text formatter, and because those few who develop new programs are careful, surprisingly few crashes occur. A future version of this operating system will be developed for a version of the PDP11 having core protection and relocation.

Virtual memory systems

One of the more interesting areas in minicomputer software is the use of virtual memory to manage the core and secondary storage of a small machine. A system in an earlier example, GRAPHIC-2, supported a form of virtual memory without the use of special hardware. Another similar example is a small time-sharing system on a Honeywell DDP516. A different example involves the use of hardware assisted segmentation and paging on a specially modified Interdata Model 3.

CONCLUSIONS

There has been a decided evolution toward greater sophistication in the design and use of minicomputer software during the relatively short years of the minicomputer era, although many of the earlier methods are still in use. A major force toward software improvement has been the introduction of new and better machines and the advent to relatively inexpensive disk storage. The expanding minicomputer field has benefited from the attraction of fresh software design talent often with large machine experience. Some recent forefront efforts are yielding minicomputer operating systems with features previously found only on large machine systems.

REFERENCES

1 Minicomputers
Auerbach Technology Evaluation Series EDP Series No 4
Auerbach Info Inc Philadelphia Pa 1971
2 D J THEIS L C HOBBS
The minicomputers revisited
Datamation Vol 17 No 10 pp 25-34 May 15 1971
3 M V MATHEWS
Choosing a scientific computer for service
Science Vol 161 p 2368 July 5 1968
4 J R COX JR
Economies of scale and specialization in large computing systems
Computer Design November 1968
5 P J DENNING
Virtual memory
Computing Surveys Vol 2 No 3 pp 153-189 September 1970
6 W H NINKE
Graphic-I—A remote graphical display system
AFIPS Conference Proceedings Vol 27 Part 1 1965 FJCC
Spartan Books Washington D C pp 830-846
7 C CHRISTENSEN E N PINSON
Multi-function graphics for a large computer system
AFIPS Conference Proceedings Vol 31 1967 FJCC
Thompson Books Washington D C pp 697-711
8 P B DENES M V MATHEWS
Laboratory computers: Their capabilities and how to make them work for you
Proceedings of the IEEE Vol 58 No 4 pp 520-530 April 1970
9 Special issue on computers in industrial control
Proceedings of the IEEE Vol 58 No 1 January 1970
10 C L SMITH
Digital control of industrial processes
Computing Surveys Vol 2 No 3 pp 211-241 September 1970
11 H E PIKE JR
Process control software
Proceedings of the IEEE Vol 58 No 1 pp 87-97 January 1970
12 D L MILLS
Preprocessors in a data communication environment
Proceeding of ACM Conference on Problems in the Optimization of Data Communications Systems
Pine Mountain Ga pp 286-310 October 1969
13 J F OSSANNA J H SALTZER
Technical and human engineering problems in connecting terminals to a time-sharing system
AFIPS Conference Proceedings Vol 37 1970 FJCC
p 355-362
14 PDP11 paper tape software programming handbook
DEC-11-GGPA-D Digital Equipment Corporation
Maynard Massachusetts 1970
15 PDP11 peripherals and interfacing handbook
Digital Equipment Corporation Maynard Massachusetts 1971
16 PS/3 system user's guide
DEC-PS-MEPA-D Digital Equipment Corporation
Maynard Massachusetts 1970
17 PDP11 disk operating system
DEC-11-SD-A Digital Equipment Corporation
Maynard Massachusetts 1971
18 C CHRISTENSEN A D HAHSE
A multiprogramming, virtual memory system for a small computer
AFIPS Conference Proceedings Vol 36 1970 SJCC AFIPS
Press Montvale New Jersey pp 885-900
19 B H LISKOV
The design of the venus operating system
Proceedings of the Third ACM Symposium on Operating Systems Principles Stanford University pp 11-16 October 1971