# Personal Distributed Computing: The Alto and Ethernet Hardware

Charles P. Thacker
Systems Research Center
Digital Equipment Corp.
130 Lytton Avenue
Palo Alto, CA 94301

## Abstract

Between 1972 and 1980, the first distributed personal computing system was built at the Xerox Palo Alto Research Center. The system was composed of a number of Alto workstations connected by an Ethernet local network. It also included servers that provided centralized facilities. This paper describes the development of the hardware that was the basis of the system.

## Introduction

In the last few years, a new type of computing environment has become available. These *distributed personal computing* systems represent another step in the process, started by timesharing, of bringing computing power closer to the user. Although many variations are possible, these systems share a number of characteristics:

- They are based on *workstations*—personal machines that are sufficiently powerful to satisfy essentially all the computational needs of a single user. The workstations include high resolution displays, and provide a highly interactive user interface.
- The workstations are interconnected by *local networks* that provide high bandwidth communication throughout a limited area, typically a single building.
- In addition to the workstations, the network contains *servers*—nodes that provide capabilities that need to be shared, either for economic or logical reasons.

Timesharing systems grew primarily from the vision of man-computer symbiosis presented by J.C.R. Licklider in a landmark 1960 paper [29]. Efforts to realize the possibilities presented in this paper occupied the creative talents of many computer science researchers through the sixties and beyond. Distributed personal computing systems build on this view of the way computers and people interact by providing a level of responsiveness that timesharing systems cannot achieve.

The first distributed personal computing system was built at the Xerox Palo Alto Research Center over a period spanning 1972 to 1980. The workstation used in this system was the Alto [36]; the network was Ethernet. This paper describes the hardware that was the foundation of this system. A companion paper by Butler Lampson [24] describes the software that was built on the hardware base described here.

This paper contains seven sections: Section 1 describes the environment in which the work was done. Section 2 traces some of the underlying ideas. Section 3 describes the early implementation period, and section 4 discusses the servers that provide printing and file storage in the system. In section 5, the reengineering effort that made the Alto into a successful internal product is described. Section 6 briefly discusses some of the Alto's successors, and section 7 contains concluding remarks.

## 1 The Environment

The Xerox Palo Alto Research Center (PARC) was established in 1970, primarily through the efforts of Jacob Goldman, director of corporate research. It was composed of three laboratories: the Computer

Science Laboratory, the System Sciences Laboratory, and the General Science Laboratory. To direct the new center, Goldman recruited George Pake, a physicist who was at that time provost of Washington University, St. Louis.

To establish the Computer Science Lab, Pake engaged Robert Taylor, who had directed the Information Processing Techniques Office of ARPA during the late sixties. Taylor had worked with and funded many of the leading computer science research groups during this period. As a result, he was in a unique position to attract a staff of the highest quality.

During the first year of CSL, Taylor built a group of approximately fifteen researchers, drawn from MIT, the University of Utah, and CMU. Several members of CSL came from Berkeley Computer Corporation, a start-up company composed primarily of individuals from the University of California at Berkeley, who had built one of the first timesharing systems [25]. From Bolt, Beranek and Newman, Taylor later brought Jerome Elkind, who was the manager of CSL from 1971 until 1976. Also during this period, Alan Kay, who was to provide much of the vision on which the Alto was based, was recruited into the Systems Sciences Laboratory by Taylor. Kay established the Learning Research Group (LRG), and defined its goal: To produce a programming system in which "...simple things would be simple, and complex things would be possible [20]."

The research environment built by Taylor was one of the main reasons for the success of CSL and its projects over the next decade. Unlike other PARC laboratories, CSL was not organized into permanent groups. Instead, researchers were encouraged to move between projects as their talents and the needs of the projects dictated. This flat structure and the mobility it made possible encouraged members of the lab to become familiar with all activities. Additionally, it provided a continuous form of peer review. Projects which were exciting and challenging obtained something much more important than financial or administrative support; they received help and participation from other CSL researchers. As a result, quality work flourished, less interesting work tended to wither.

During 1971 and early 1972, most of the effort in CSL was spent in building a set of hardware and software facilities to support the future work of the laboratory. The Maxc timesharing system [13] was built and the Tenex [2] operating system was acquired and modified for it. Projects in graphics, computer networking, and language design were started.

The main research theme of CSL—office information systems—was also developed during this period. Most of the research done in CSL and SSL during the next five years was organized around this theme, which reflected the desire of Xerox to expand its traditional copier business to include most of the functions performed in offices. Eventually, this theme was broadened to include what is now known as distributed personal computing, but initially our ambitions were lower.

A strategy for carrying out work in experimental computer science was also adopted at this time. It was based on the idea that demonstrations of "toy" systems are insufficient to determine the worth of a system design. Instead, it is necessary to build *real* systems, and to use them in daily work to assess the validity of the underlying ideas, and to understand the consequences of those ideas. When the designers and implementors are themselves the users, as was the case at PARC, and when the system is of general utility, such as an electronic mail system or a text editor, there is a powerful bootstrapping effect. This view of systems research is quite different from that found in most academic environments, since it requires larger groups working over a longer period than a university can usually support.

It was clear that the ability to provide systems with high levels of functionality would be limited by software considerations far more than by the capabilities of the underlying hardware. While it would be necessary for us to build hardware, since the needed capabilities were not commercially available, the characteristics of the devices would be determined primarily by the needs of the software systems for which they were intended. This view is commonplace today, as hardware performance has increased and its cost has declined, while the cost of delivering large, reliable software systems has continued to increase. It was much more radical in 1971.

Although hardware development was an integral part of the work carried out in CSL, it represented a small fraction of the overall activity of the laboratory. At no time did the number of people engaged primarily in hardware work exceed five—roughly ten percent of the total professional staff. This core group was very effectively augmented during large projects (e.g., the development of the Dorado [27] during 1977-80) by laboratory members with computer science, rather than electrical engineering backgrounds. However, most things were done by a relatively small group. For this reason, it was necessary to be selective in our choice of projects. Simplicity and utility were the most important criteria. Highly complex designs would have been beyond our capabilities, and the construction of systems without a wide user community would not have justified the expenditure of scarce design and implementation talent.

## 2 Sources of Ideas

By late 1972, most of the laboratory facilities were in place in CSL, and the researchers who had produced them began planning longer-term projects. It was at this time that the main ideas underlying the Alto were discussed and refined into an actual design. Although a number of people in CSL and SSL contributed to the specification of the new system, Butler Lampson, Alan Kay, and Bob Taylor were the individuals who were primarily responsible for shaping the design. To the extent that CSL had project managers, I filled that function. My task was to convert the vision of Lampson, Kay, and Taylor into working hardware.

Taylor had originally proposed that the primary computing facility in CSL should be an interconnected collection of small display-based machines, rather than a central timesharing system. He thought that a sufficient amount had been learned about the design of interactive systems, and that hardware costs were low enough that it would be feasible to produce such a system in modest quantities. In 1970, this idea contained a number of technical difficulties. Lampson and I argued that a stable set of computing facilities as well as experience in producing hardware in the new laboratory would be required before embarking on such an ambitious project. This conservative view prevailed, but by 1972, sufficient progress had been made in a number of areas, particularly semiconductors, that the difficulties did not seem as overwhelming.

In his 1969 thesis, Alan Kay had described a small computer system, the "reactive engine" [21], that shared many of the characteristics of the new machine. Like Taylor, Kay wanted a system that would provide a complete work environment for its user, including text and graphics manipulation, computing, and communications capability. By 1972, this vision of computing had acquired a name—Dynabook—and for a while, the Alto was called the "interim Dynabook" by its developers. Kay's vision was that the ultimate Dynabook would be portable, so that it could provide all the functions provided by books, paper, pencil, and terminals. Although the Alto never achieved this part of his vision, it served for a number of years as the hardware environment for the Smalltalk system [16], in which a number of text and graphics, music, and simulation applications were built.

Lampson's view of the capabilities of the new system and its uses was perhaps the most explicit. In a 1972 guest editorial [23] in Software–Practice and Experience, he had predicted that within five years, it would be possible to build a system "...comparable to a 360/65 in computing power for a manufacturing cost of perhaps $500." In the same article, he predicted significant advances in programming technology, and foresaw the some of the effects that would follow from these developments:

> As a result, millions of people will write non-trivial programs, and hundreds of thousands will try to sell them... Almost everyone who uses a pencil will use a computer, and although most people will not do any serious programming, almost everyone will be a potential customer for serious programs of some kind.

Although it was clear that we could not achieve these goals in 1972, there was a clear consensus that the new system should have characteristics that were prototypical of this vision. Cost was not a primary consideration in the design, but it could not be outrageous, since the system had to be producible in modest quantities to justify the development of software for it.

By late 1972, the principal features of the new machine had been defined. The major departure from past systems was the machine's display (see Figure 1). To emulate as many of the characteristics of paper as possible, we chose to provide a full bitmap, in which each screen pixel was represented by a bit of main storage, and to use raster scanning rather than the lower-cost calligraphic techniques popular at the time. We were encouraged by the earlier experiences of a group in SSL, which had developed a character generator for a similar, but higher resolution display. The display resolution was 606 pixels horizontally by 808 pixels vertically, which allowed display of a full page of text. The display image was refreshed directly from main memory, so arbitrary graphics could also be produced using the machine's load and store instructions. Initially, a specialized instruction was provided to paint characters from a font in memory into the bitmap; this facility was later superseded by the more general BitBlt primitive invented by Dan Ingalls [17]. The Alto contained no support for other graphic primitives, since we were primarily interested in text and engineering drawing applications that could be done with specialized character sets and straight lines.

The decision to provide a high-performance display came directly from our view that the most important purpose of the machine and the software that would be written for it was to provide a high bandwidth interface to the human user. Timesharing systems had made computing more accessible and decreased its

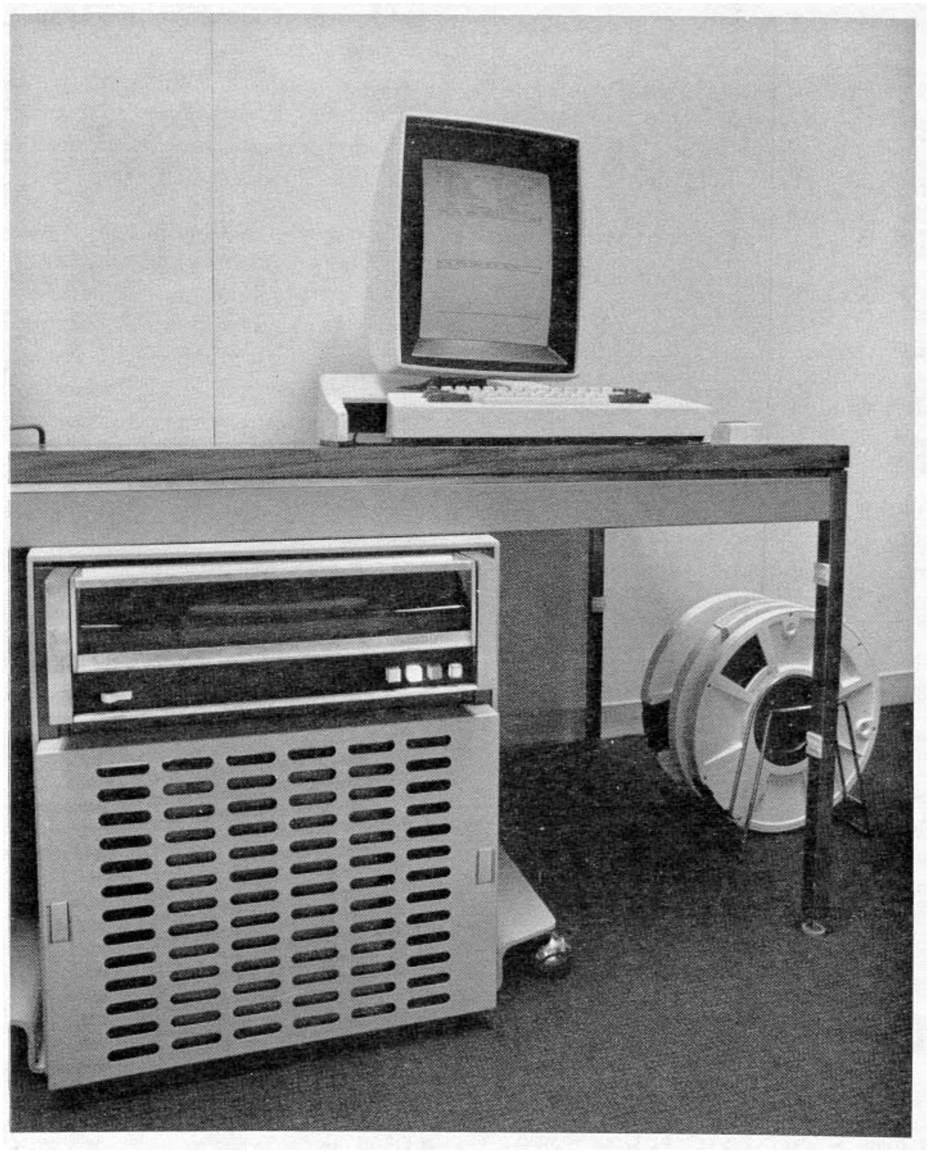

Figure 1: The Alto II Workstation. The Alto I had an identical display, keyboard, and mouse, but a slightly different cabinet.

cost, but they had done little to increase the *quality* of man-machine interaction. We viewed improvement of the user interface as extremely important, and were willing to expend a considerable fraction of the machine's resources in providing it.

Another important feature of the user interface provided by the Alto was the use of a mouse as a pointing device. This was not a PARC innovation; it had been used with considerable success in the pioneering NLS system of Englebart done at SRI in the late sixties [12,11,10]. When rolled over a work surface adjacent to the keyboard, the mouse provides relative positioning information, usually used by software to control the position of a cursor on the display screen. It also provides additional input through buttons on its top surface. Subsequent research [5] has shown that the mouse is a Fitts' Law device, in that it is as efficient for target selection as manual pointing. The practical impact of this is that in the domain for which it was intended, the mouse, like the compact disk in the audio domain, does as well as the limits of the human user allow. This is often overlooked by those attempting to provide better pointing devices. The mouse is not as effective as a pencil or a graphics tablet for freehand drawing, but very few graphic applications made use of a tablet, although an interface for one was provided.

The display's cursor was a small image, sixteen pixels square, whose contents and position could be controlled by software. Many programs made considerable use of the programmability of the cursor, using its contents to convey information about the item to which the user was pointing.

The Alto keyboard was similar to that of a typewriter; it was not accidental that it lacked the cursor positioning keys and numeric keypad found on most personal computers today. In addition to the normal typing keys, it provided eight uncommitted keys that could be used by software as option or function keys. A five-finger keyset, which had been used successfully in Englebart's NLS, was provided as an enhancement to the keyboard, but it required a trained operator for use, and never became popular as an input device.

The original Alto contained 128 thousand bytes of main storage, and a 2.5 million byte cartridge disk. This was similar to contemporary minicomputers, and constituted a fairly serious error. If we had understood how rapidly semiconductor technology would advance, and how long the Alto would live, we would have included more convenient means for accessing a larger memory. We failed to do this, and although the main memory was subsequently expanded to 512 thousand bytes, it was difficult for programs to make use of the additional memory. This hampered software development quite a bit in later years.

The processor of the Alto was specified with flexibility and expansion in mind. It was microcoded, which allowed us to experiment with new instruction sets and with with new input-output devices. The principal characteristic that served to differentiate it from the minicomputers of its time was that the microprogrammed processor was shared between the emulation of a target instruction set and the servicing of up to fifteen additional fixed-priority *tasks*, most of which were associated with the machine's input-output devices. Task switching occurred rapidly, typically every few microseconds. This mechanism allowed the input-output controllers to be very simple, since they could make use of the processor for much of their work. Since access to the single-ported memory is the bottleneck in a small system, multiplexing the processor in this way did not degrade system performance. This technique had been used before on the Lincoln Laboratory TX-2 [14]; it was very successful in the Alto, and has been used in several of the Alto's successors.

To be an effective replacement for centralized computing facilities, personal workstations require a means for communicating with other nearby workstations and with servers that provide shared facilities such as file storage, printing, and long-haul communication. When the Alto design was started, we realized that such a facility would be needed, but did not understand its requirements well enough to begin a design. During late 1972 and early 1973, a number of alternatives were considered, ranging from star-connected serial links operating at a few hundred thousand bits per second to a parallel bus scheme operating at several million bytes per second. The need for bulk file transfers ruled out the low bandwidth of the first approach, and the complexity of the required cabling made the parallel bus unattractive. Coaxial cable, connected with standard cable television components, was tentatively selected as the transmission medium, since it would meet both the bandwidth and reliability requirements.

Several transmission methods were also considered, and a variant of the packet-based Aloha [1] radio system was selected. In pure Aloha, stations needing to transmit packets simply do so, and the resulting interference between stations reduces the channel capacity considerably. We realized that better performance was possible, since our cable Aloha stations could detect when their own transmissions were being interfered with, and abort them without transmitting a complete packet. We chose a baseband, as opposed to a carrier system because it is considerably simpler, and the extra bandwidth afforded by the latter scheme did not appear to be needed. These tentative decisions about the form that the network would take were made in late 1972, but little progress was made on an actual network design until Bob Metcalfe, who had joined CSL in mid-1972, and David Boggs, who came to SSL in March, 1973, began working on what was to become the Ethernet [31].

## 3 Implementation

In November 1972, implementation of the Alto began. The design was completed in approximately two months, including an initial version of the microcode for an instruction set emulator and for the display and disk controllers. Two prototype machines were then built using wirewrap technology, and were in operation in April 1973. One reason for the short schedule was that we had developed a rapid prototyping facility as a part of the construction of the Maxc timesharing system during 1971 and 1972. We were also able to use the memory boards originally developed for the Maxc system in the Alto, which saved considerable engineering effort. The design of the processor, memory subsystem, and display controller was done by Chuck Thacker, while the disk controller and its microcode were designed and im-

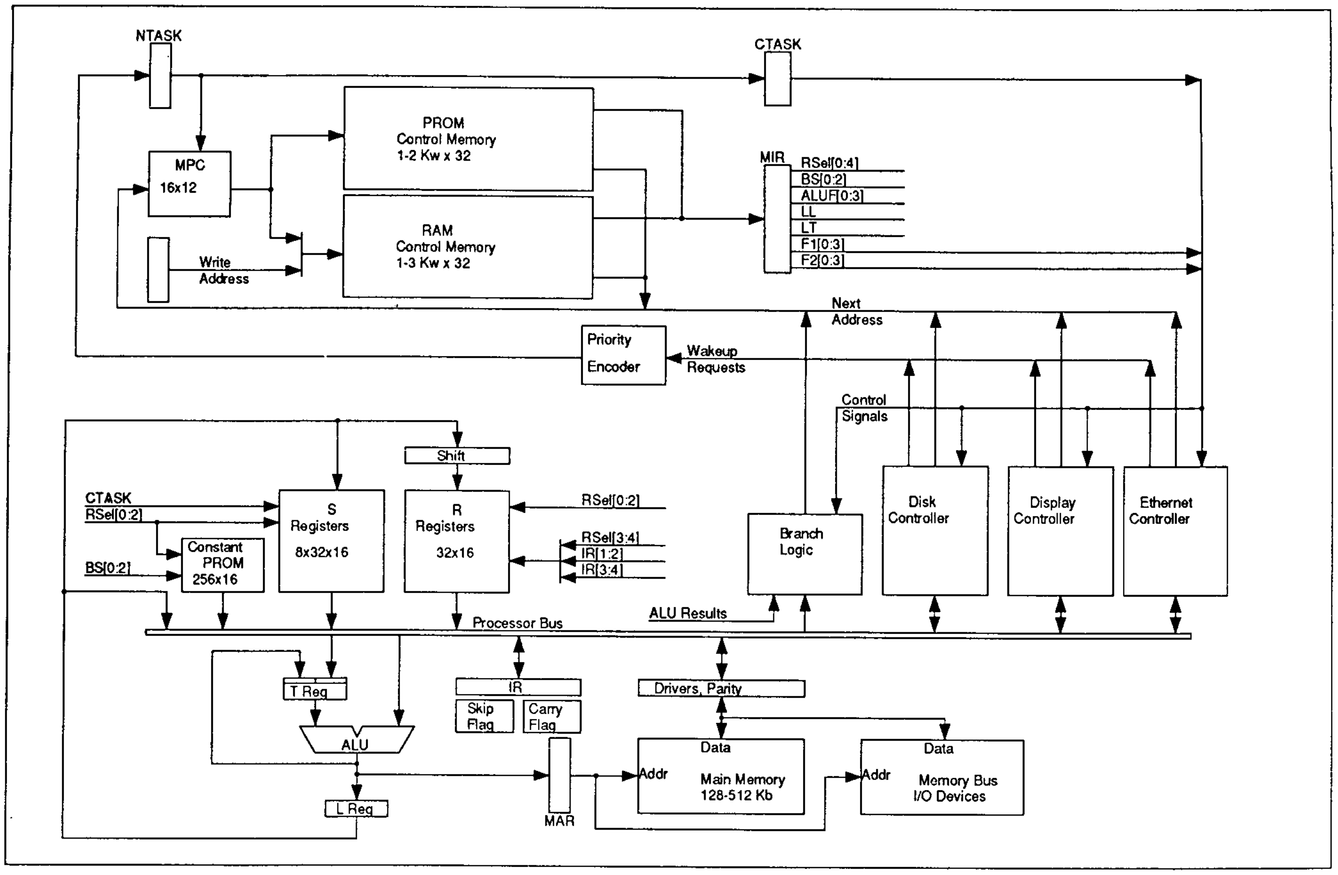

Figure 2: The Alto Microprocessor.

plemented by Ed McCreight. Larry Clark built the early systems and designed the package.

The Alto was a very simple machine by today's standards. The processor (Figure 2), is composed of three printed circuit boards containing about 200 small and medium-scale integrated circuits. Each of the input-output controllers occupies a single board containing approximately sixty integrated circuits. The processor is organized around a 16-bit bus connected to the main memory, an arithmetic unit, a number of high speed registers (R and S), and the input-output controllers. The transfer of data over the bus, the operations to be performed on the data, and the tests to be applied to it are controlled by a 32-bit microinstruction taken from PROM or RAM control store. Microinstructions are executed by a two stage pipeline which can start a new instruction every 170 nanoseconds. The processor is shared among sixteen fixed priority tasks. The NTASK and CTASK registers hold the number of the task currently in control of the processor. NTASK addresses the sixteen-element MPC RAM, which holds the task program counters. NTASK is loaded with the number of the highest priority wakeup request whenever the running task is willing to relinquish the processor. The only state associated with a task that is saved by the hardware is the task's program counter. Other machine resources are shared among tasks by programming convention. Normally, a task switch takes place with no overhead, unless it is necessary for the task giving up control to save and restore the L or T register.

The Alto main memory is synchronous with the processor, which starts all references by explicitly loading the memory address register (MAR). The memory can read or write a single 16-bit word in five machine cycles, or it can read a 32-bit doubleword in six cycles. The doubleword read was originally provided to support the display, which consumes two-thirds of the memory bandwidth even with this operation; it was also used very effectively by instruction set emulators for instruction fetching and to manipulate 32-bit quantities.

The Alto was not a high-performance machine, even by the standards of its time. Without the performance degradation caused by the input-output devices, it requires between one and three microseconds to execute a single emulated instruction. With the display running, these times are increased by a factor of three. Until software was developed that required a great deal of computation for simple user actions (e.g., the Bravo text editor), the speed of the machine was adequate. Perhaps more important than the absolute speed is the fact that the performance of the Alto is *predictable*. It is very difficult to provide this characteristic in a timesharing system, and its lack can be very disconcerting to the user. The Alto cannot provide the peak performance of a timesharing machine, but it has the desirable property, pointed out by Jim Morris, that it doesn't run faster at night.

Once the design was complete, the microprocessor was simple enough that the hardware worked almost immediately. Debugging the microcode was more difficult, but was simplified considerably by an auxiliary writeable microstore built for the purpose. This device was connected to the control logic of the Alto under test. It replaced the PROM control store with RAM, and also added several bits to the microword. These additional bits were used to provide a breakpoint capability that made debugging much easier. The test unit was under control of a dedicated minicomputer that ran a microcode assembler and debugger. Using these tools, microcode debugging could be carried out as easily as debugging an assembly language program on a conventional machine.

The first of several microcoded instruction set emulators developed for the Alto was done for a virtual machine similar to the Data General Nova minicomputer. We had previously purchased a number of these systems, and had developed for them a compiler for BCPL [32], a predecessor of the popular C language. The main differences between the Alto instruction set (Figure 3) and that of the Nova were that the size of the Alto's address space was twice that of the Nova, and a number of instructions were added to support the Alto's input-output and interrupt system and to optimize BCPL procedure calls.

The resulting instruction set was not compatible with that of the Nova, but it was sufficiently similar that modifying the compiler was straightforward. Most of the early software for the Alto was written in BCPL. Only a small amount of assembly code was ever written for the machine. The microprocessor hardware contained a small amount of logic to enhance the performance of the BCPL emulator. This logic included a register to hold the instruction being executed and a method to address the registers used for the emulated machine's accumulators from its fields. This logic was not used by later emulators, and it probably would have been better to have used the same amount of hardware to provide functions with more general utility.

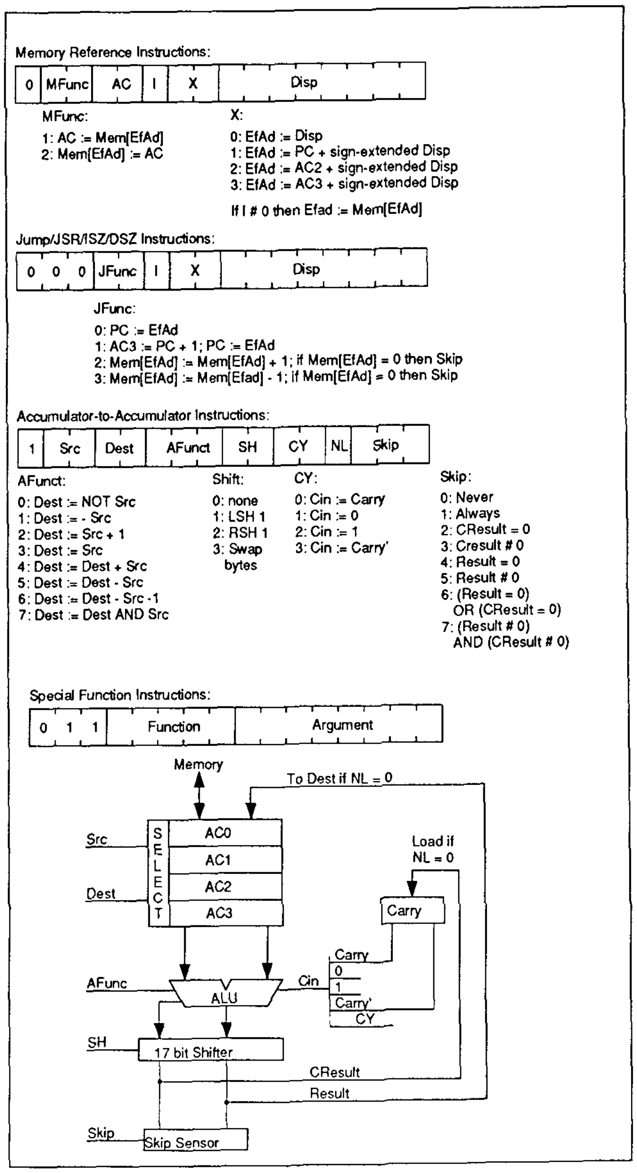

Figure 3: The BCPL Instruction Set and Processor Model

Emulators for several other instruction sets, including Smalltalk [18,16], Lisp [8,7], and Mesa [15] were written for the Alto. All were based on the idea of encoding the instructions as *bytecodes*. This arrangement allows instruction decoding to be done by a single 8-bit dispatch, rather than the several dispatches required to decode a BCPL instruction. The performance gained in this way more than offsets the lack of specialized decoding hardware. Most of the software written for the Alto after 1977 used Mesa, which was the choice in CSL, or Smalltalk, which was the language of choice in LRG. The Alto Lisp system was unsuccessful, primarily due to the lack of sufficient memory.

Input-output devices can be connected to the Alto in several ways. High performance controllers that make use of the microprocessor or require a high bandwidth connection to the memory are connected directly to the processor bus. The display, disk, and Ethernet controllers are examples of devices in this class. Lower performance devices are attached to the memory bus and addressed as locations at the end of memory, as in the PDP-11. The keyboard and keyset are connected in this way. Finally, a parallel port is provided for low performance devices outside the cabinet. This port was used to connect a variety of devices, including impact printers, a PROM programmer, and X-Y digitizing tablet, and a cassette tape deck.

The hardware controllers for the disk, the display, and the Ethernet are similar. They contain data buffering, logic to drive and receive control lines required by the device, and a small amount of device-specific timing and control logic. Most of the complexity associated with a controller is contained in its microcode. The display, for example, makes use of three of the Alto's sixteen microtasks: one is awakened during the display's vertical synchronization period, one is awakened during horizontal synchronization, and one is responsible for transferring raster data from main memory to the controller data buffer. The controller hardware provides wakeup requests at the appropriate times, but the microcode is responsible for carrying out most of the work required to maintain the display.

This arrangement, in which a device controller has the full computational power of the processor at its disposal, allowed us to provide convenient logical interfaces between devices and the driver software that operated them. The display interprets a chain of control blocks in main memory, each of which defines the contents of a horizontal band on the display screen. Since areas of the screen containing only white space, such as the space between lines and paragraphs, are not required to have any underlying bitmap memory, this technique reduces the display storage required by the standard text editor from the 61 thousand bytes required to represent a full screen to approximately 50 thousand bytes, a substantial saving in a machine with only 128 thousand bytes of memory. Drawing programs that allow the user to manipulate full-screen images cannot take advantage of this economy, but such programs are considerably simpler and smaller than the editor, so efficient use of space is less important in these applications.

The disk controller also makes extensive use of the Alto microprocessor. Like the display, it executes a chain of control blocks from main memory. Each control block specifies a 512 byte page to be read or written. The controller is able to transfer consecutive physical sectors between the disk and memory, which represented unusually high performance for the time. The controller uses two of the Alto's microtasks, one of which is awakened every sector, and one which is responsible for data transfers within a sector.

The disk controller and the file system were designed concurrently by Ed McCreight and Butler Lampson, with reliability as a primary goal. The designers wanted to provide a system in which hardware or software errors would result in a minimum amount of information loss. An important innovation—the use of label blocks—contributed substantially to the system's reliability. Labels add a third record to the header and data records customarily contained in a disk sector. The label record contains a unique integer identifier for the file containing the sector, the sector's position in the file, the addresses of the previous and next pages of the file, and the number of valid bytes in the data record. The controller microcode checks the information in the label block before doing any operation on the data record. This check ensures that the disk is properly positioned, both physically and logically, before any access to the data is done.

The use of label blocks, combined with replication of directory information in the first page of every file, means that the directory, which is an ordinary file, can be reconstructed from the contents of the disk itself if it becomes scrambled. Similarly, if data pages are corrupted, it is possible to determine precisely the extent of the loss, and to preserve the balance of the file system. One of the earliest programs written for the Alto was the Scavenger, which verified the integrity of a file system and corrected inconsistencies. This program, the first version of which was written by Jim Morris, makes the loss of even a small portion of a file system an extremely rare event.

The original Alto prototypes did not contain Ethernet interfaces, but during 1973, substantial progress on both network hardware and protocols was made. The name "Ethernet" was first used in May 1973 by Bob Metcalfe. Metcalfe and David Boggs worked on the network facilities during the summer and fall, and the prototype machines were exchanging packets by the end of 1973. Although the Alto was the first machine equipped with an Ethernet interface, Metcalfe and Boggs went on to design controllers for all the PARC computers, including Data General Novas and DEC PDP-11s. The Ethernet transceiver, an analog device that connects the controller to the coaxial cable, was designed by Tat C. Lam. The original Ethernet was slower than the commercial version available today. Its transmission rate was 2.94 million bits per second, half the rate of the Alto master clock, and it used Manchester encoding for the serial data. For collision detection, it relied on comparison between the actual signal on the coaxial cable and the signal the transmitting station was attempting to send, rather than the level monitoring done by the commercial version. The 3 million bit per second bandwidth of the network was of some concern initially, in that we were not sure that it would be sufficient for a large system. Subsequent measurements [33] of a large network revealed that loads in excess of ten percent of full capacity were rare. There might have been less excess capacity if diskless workstations that paged over the network had been employed, but the use of local disks made this unnecessary.

In April, 1973, the first Alto prototype was completed. It was able to run simple programs to exercise the disk and display. The first image to be displayed was the Sesame Street 'Cookie Monster', which had been carefully digitized by a member of Kay's group.

During the balance of 1973, nine more prototype machines were built at PARC. During the summer, the prototypes had been tested sufficiently that we were willing to commit the design to printed circuit boards. At the same time, we realized that the lack of writeable control store would be a serious limitation in a machine intended for experimentation, and added one thousand words of instruction RAM to the original PROM control store. The original microcode was improved substantially, and a number of test programs for the hardware were written. Software development had begun, but there was not enough software available by the end of 1973 for the Alto to replace our timesharing system as the main computing facility for users.

One of the major strengths of the Alto in a research environment was that it could provide very high performance if the user were willing to accept the unpleasantness of microprogramming the processor directly. The first demonstration of such an application took place in the summer of 1973. Alan Kay, who was an accomplished organist, wanted a synthetic instrument with natural sound quality. He believed that this could be accomplished by recording and digitizing a real organ, and doing table lookup to provide a completely faithful replica of the waveshape. We decided to give him his wish, and purchased a two-manual organ keyboard (with pedals), a precision digital-to-analog converter, and a high fidelity amplifier and loudspeaker system. The keyboard and DAC were interfaced so that a task-specific function could read the keyboard as a bit vector, and load a value into the DAC. The organ simulator kept in main memory a table, consisting of the amplitudes at 256 points along a single cycle of the desired waveshape. Samples were generated for the DAC at a 25 kHz rate by taking points from this table at offsets which were inversely related to the frequency to be generated. The microprogram computed up to ten such samples every 40 microseconds, one for each key that was depressed. Samples were summed and the result was transmitted to the DAC. A variety of different effects could be generated by using different variations of the 'canonical cycle'. Although this application was fairly frivolous, it was an impressive demonstration of the real-time capability of the machine. Later, a serial line concentrator that connected up to sixty-four serial 300 baud lines to the Ethernet was built using similar techniques. The only extra hardware used in this device was a group of level converters and latches to allow the microprogram to read the value of the received data and store the data to be transmitted. All other processing was done by specialized microcode.

By late 1973, we were ready to produce a quantity of the machines for CSL and SSL. Although we had been able to build a small number of prototypes, the manufacture of the thirty systems we needed was beyond our capabilities. Fortunately, the company had established a custom systems manufacturing group in Los Angeles, and this group agreed to manufacture the Alto for us. The first machines were delivered between May and September, 1974. This was only slightly later than the first release of the basic Alto software, which took place in March, 1974. This early software consisted of the operating system [28], the BCPL compiler, and a primitive text editor. It allowed many of the researchers in CSL and SSL to begin doing a substantial amount of their work on the Alto, although the Maxc timesharing system was still used for electronic mail, file storage, and printing.

## 4 Servers

The most important components of a distributed computing system, after the workstations and the network that interconnects them, are the servers that provide shared facilities. We initially underestimated the importance of servers, assuming that the facilities provided by a set of workstations would be sufficient. We soon discovered that this was incorrect. Some functions, such as high-quality printing, are very expensive, and must be shared for economic reasons, while in other cases, sharing is used to provide communication between the users of the system. File storage is an example of the latter situation, although the low cost per byte of large disk files also provides economy of scale.

The first PARC server was EARS, a printing server named after its components: Ethernet, Alto, RCG (research character generator), and SLOT (scanning laser output terminal). This system was quite successful, in spite of the considerable obstacle to its acceptance presented by its bizarre name. It was the forerunner of the Xerox 9700 printer, which has been an extremely successful product.

The printing portion of EARS had been under development even before PARC was founded. Gary Starkweather, an optical engineer at the Xerox Webster Research Center, joined SSL in 1971. He brought with him a prototype printing engine consisting of a laser scanner attached to a standard Xerox copier. This device used a rotating polygon to scan an intensity-modulated laser beam across a standard xerographic drum, building up a raster image of the page being printed. During 1972, Ron Rider of SSL and Butler Lampson designed and implemented a character generator capable of printing high quality text in several fonts on Starkweather's engine, at a speed of one page per second.

A printer based on these components, driven by a Data General Nova minicomputer, was demonstrated in late 1973, but was never placed in service. Instead, Rider decided to build an Ethernet-based print server. He modified the character generator to allow it to be driven from an Alto, and wrote the necessary software to control the printer. The network-related portions of the server were written by Bob Metcalfe. EARS was the first major application for the Ethernet, and during its development, several experimental communications protocols [30] were tested and refined. EARS was placed in service in the Fall of 1974, and provided printing service to CSL and SSL until it was replaced by the Dover printer in 1977.

The final component needed for a complete distributed computing system was a file server. The hardware basis for several experimental file servers was a high-performance disk controller designed by Roger Bates in 1975. Like earlier Alto device controllers, it made use of the Alto's microprogrammed tasking for many of its functions, but it was considerably more complex than the earlier controllers because of the high bandwidth of the attached disks. Using this device, it was possible to connect as many as seven 300 million byte disks to a single Alto.

The first file server, Juniper, was to have provided page-level access to files, as well as atomic transactions. Planning for Juniper began in 1974, but actual programming did not begin until 1976. It was completed in 1977, but was never used extensively because of its poor performance.

The file server that was used most widely was the "Interim File Server", or IFS. This software was written in 1976 by Ed Taft and David Boggs, when it became clear that Juniper would not be ready as early as originally anticipated. It was an extension of the simple Alto file system, combined with the PUP (PARC Universal Packet) file transfer protocol [3]. It provided only bulk file storage, but it was completed rapidly and was reliable and efficient. However, as an "interim" system, it was a failure, since the IFS still provides the majority of the file storage in the Xerox internetwork.

The most complex server built using the Alto was the Dover printer, designed in 1976. Dover used a raster printing engine that was a descendant of the SLOT used in the EARS server, driven by a controller that relied heavily on the input-output processing capability of the Alto. Dover was a large project, involving several groups within the company. The design of the printing engine was done by the Special Projects Group (SPG) in Los Angeles, the group that manufactured the Alto. John Ellenby of CSL was responsible for management of the engine development. Software for the Dover was written by Dan Swinehart of SSL, and Bob Sproull of CSL. The development of the controller was done by Severo Ornstein of CSL, Bob Sproull, and Jim Leung of SPG, from a design by Butler Lampson.

The controller was considerably simpler than the earlier EARS character generator. Instead of using hardware to generate the bit stream for the printer in real time, the controller built up the image to be printed incrementally. Two buffer memories were used, each capable of holding sixteen of the printer's scan lines. While one of these buffers was being serialized and sent to the printer under hardware control, the other was being loaded with video information. The text to be printed and the bitmap representations for the fonts to be used were kept in Alto main memory, and written into the buffer by a high priority microcoded task. This process was then repeated for each of the roughly 250 bands that made up each page. Spooling of files received from the network was done by a BCPL program, which also sorted the contents of each page into bands in preparation for transmission to the controller. The resolution of the Dover was 384 pixels per inch, lower than the 500 pixels per inch of EARS, but still adequate for text and line graphics. The bandwidth requirements of the printer were high enough that the machine could not receive and spool files while printing, but its printing speed of one page per second was high enough that this was not a problem, since it could stop between pages to receive files. Incremental image generation also placed a limit on the complexity of pages that could be printed. For the few images that exceeded the limit, another server that composed an entire page on disk and transmitted it to a slower printing engine was available.

The controller contained approximately 300 integrated circuits, making it about one eighth as complex as the character generator used in EARS. This simplification was made possible by the extensive use of the Alto's microprocessor to provide the low level control functions for the interface. Several dozen Dovers were built, and a number are still in operation.

## 5 Expansion

During 1975, the Alto was redesigned to improve its reliability and reduce its cost. The work was done by the Special Projects Group in Los Angeles that had been producing Altos for PARC, and was planned and supervised by John Ellenby of CSL. As part of this effort, all the boards and the package were redesigned. The memory system was reimplemented using 4 thousand bit RAM chips, and error correction was added. The resulting machine was much easier to build and service than the earlier system, and its cost was much lower—approximately twelve thousand dollars, rather than eighteen thousand dollars. The Los Angeles group had built a total of sixty of the original systems over a period of two years, most of which had been purchased by CSL and SSL. They were now able to produce the system in high volume. Over the next four years, approximately fifteen hundred Altos were built, of which approximately a thousand are still in use today. Most are used by individual engineers and scientists in a number of Xerox facilities, although many were configured as servers, and a few were used for marketing probes or donated to university computer science groups.

A final redesign done in 1979 replaced the memory with sixteen thousand bit RAM chips, and increased the amount of memory that could be attached to 512 thousand bytes. At the same time, the microcode store was changed from one thousand words of RAM and two thousand words of PROM to one thousand words of PROM plus three thousand words of RAM. By this time, other language emulators had almost totally superseded the original BCPL emulator. These emulators were usually loaded at bootstrap time or as part of starting a program that used a particular language. The additional RAM control store made it possible to spend less time minimizing the space required by the microcode and concentrate instead on its functionality and performance. The microcode was improved somewhat after the final redesign, but by 1980, most new development had shifted to the Alto's successors, described in the next section.

Although most of the software developed in CSL and SSL was distributed with the Alto, three application programs were primarily responsible for the machine's popularity with technical professionals in Xerox. The Bravo editor [22], designed by Charles Simonyi and Butler Lampson, and implemented and improved by Simonyi and others from 1974 through 1978, was the first and most important of these. Bravo was the first WYSIWYG (what you see is what you get) editor. It supported multiple fonts, and its high quality output could be printed on one of the many Dover printers that were available throughout the company.

The second important application program was the Laurel mail system [4], written by Doug Brotz, Roy Levin, and Mike Schroeder in 1978. Electronic mail has a profound effect on communication within an organization, since it combines the permanence and precision of memos with the speed of the telephone. By 1980, the Xerox internetwork, composed of local Ethernets connected by telephone lines, had been expanded to most of the engineering and research sites within the company. Laurel was rapidly adopted by a large fraction of this community, and by 1983 there were over four thousand regular electronic mail users in Xerox.

The third popular application was a group of tools for digital logic design, including the SIL illustrator [35] written by Chuck Thacker in 1975, and a routing program for prototypes written by Ed McCreight. These programs also produced high quality documentation on the Dover printers. They increased the productivity of designers significantly, and are used by most of the electronic engineers in the company.

## 6 Successors

The Alto was only the first of several personal workstations built at Xerox. The Dorado [26,6,27] and the Dicentra were developed at PARC, and the Dolphin and the Dandelion were commercial systems designed in the Electronics Division and the System Development Division.

Dorado is the largest hardware engineering project ever undertaken by the Computer Science Laboratory. It was difficult to think of the Dorado as a personal machine, since it consumed 2500 watts of power, was the size of a refrigerator, and required 2000 cubic feet of cooling air per minute (while producing a noise level that has been compared to that of a 747 taking off).

It was *used* as a personal machine, however, and supplied computing power comparable to three VAX-11/780s. This may seem profligate, but it was consistent with the view that the CSL hardware base should be equivalent to that which would be commercially available and affordable in five to ten years. With hardware that is not limiting, it is possible to explore ways of using computers that are considerably ahead of current practice.

The Dorado project was started in CSL during 1975. It was moved to the System Development Division in 1976, but returned to CSL in 1977 when it became clear that the machine's high cost would not meet SDD's needs. The initial design was completed in late 1978, and two prototypes were built. A redesign, completed in 1979, was then done to simplify the machine. Manufacture of the machine started in 1980 in a small production facility that had been established for the purpose. By 1982, approximately thirty systems had been built, and the Dorado had replaced the Alto as the principal computing vehicle in CSL and SSL.

The Dorado achieved its high performance through its aggressive technology and a great deal of attention to detail on the part of its designers. It uses emitter-coupled logic (MECL 10K) with two to four nanosecond gate delays. The processor is microprogrammed, and like the Alto employs multitasking at the microcode level to operate input-output controllers. Unlike the Alto, it has virtual memory, an eight thousand byte cache, and a separate instruction fetch unit associated with the CPU. Up to 16 million bytes of main memory may be attached to the Dorado. The memory bandwidth available for input-output devices and to service cache misses is 66 million bytes per second. The processor executes microinstructions in a three-stage pipeline that can start a new instruction every sixty nanoseconds. The separate instruction fetch unit allows many instructions in the common emulators to execute in a single microinstruction.

In terms of man-years expended on a single project in CSL, the Dorado is second only to the Cedar programming environment [9,34], which its high performance made possible. The initial design was done by Butler Lampson and Chuck Thacker; the design was continued in SDD by Thacker, Brian Rosen, Don Charnley, and Tom Chang. When the project returned to PARC, it was partitioned into a number of subsystems: Ed McCreight and Severo Ornstein were the project managers; Butler Lampson was the technical leader of the project. The microprocessor was designed by Ken Pier, Roger Bates, and Ed Fiala. The instruction fetch unit was designed and implemented by Severo Ornstein, Gene McDaniel, and Will Crowther. The storage system was done by Doug Clark, Ed McCreight, and Ken Pier. A number of individuals produced the microcode for the machine, including Ed Taft, Peter Deutsch, Willie-Sue Haugeland, and Nori Suzuki.

The Dolphin was a much less ambitious successor to the Alto, designed in the Electronics Division of Xerox in 1977-79. In a sense, it was the successor of the Dorado, rather than the Alto, since it was done by the same group (Thacker, Charnley, Rosen, Chang) that had worked on the Dorado in SDD as well as a group in Los Angeles that included Jack Cameron, Howard Kakita, and Malcolm Thomson. Dolphin employed a number of ideas that had been incorporated into the Dorado, including virtual memory and a high-bandwidth input-output system. Its technology was not as aggressive as that of the Dorado—Schottky TTL, rather than ECL— and it was smaller and much less expensive. The Dolphin was used as the processor in the Xerox 5700 Electronic Printing System, and a version configured as a Lisp workstation became the 1100 Scientific Information Processor. Although PARC built approximately fifty Dolphins for internal use, and provided emulators for the Alto instruction set, Mesa, and Lisp, the machine was not popular. It had a higher resolution display, more memory, and a larger disk than the Alto, but it was only about twice as fast. Dolphin became available slightly before the Dorado, but the performance of the latter machine made it much more attractive, particularly for a research environment.

The Dandelion [19], known commercially as the Star 8010 workstation, was implemented in 1979 and 1980 by a group in SDD consisting of Bob Belleville, Robert Garner, and Ron Crane. Dandelion was based on a paper design called Wildflower done by Butler Lampson and Roy Levin of CSL. It was intended to have high performance and extremely low cost, but limited configuration flexibility. The Dandelion CPU used 2901 bit-slice processors, and employed a fixed time-slice form of multitasking that was quite different from that of the Alto. Dandelion was the first of the Alto's descendants that did not provide an emulation mode in which Alto software could be run. It was programmed exclusively in Mesa and the extended Mesa developed for the Cedar system. Dicentra, built by David Boggs and Hal Murray in 1982, was a variant of the Dandelion which included a Multibus rather than a proprietary bus for the attachment of input-output devices and memory. It provided a low-cost way to obtain a Mesa-compatible processor to which industry standard peripheral controllers and devices could be attached.

In addition to its direct descendants at Xerox, the Alto has inspired a number of similar systems from other commercial vendors. The Apple Lisa and Macintosh are perhaps the most familiar of these; Table 3 in Lampson [24] lists several others.

## 7 Conclusion

The Alto is small and slow by today's standards. The four generations of memory and microprocessor development that have passed since 1972 have made it straightforward to build low cost personal workstations with a hundred times the memory capacity and ten times the speed of the Alto. It seems likely that progress in semiconductors will continue at its present rate for perhaps another decade before fundamental physical limits are reached, so much more powerful systems are inevitable.

Higher bandwidth networks have also become much easier to engineer with the advent of fiber optics. However, experience with the Ethernet indicates that even with very high performance machines such as the Dorado, network bandwidth is not the limiting factor in overall system performance.

A surprising fact that has emerged from the work on the Alto and its successors is that the amount of software required to support interactive user interfaces is much greater than originally anticipated. Invariably, the complexity of the software is much greater than that of the hardware on which it runs. Except in a few applications in which the users are experts (e.g., programmers using programming environments), it has not yet become possible to provide the kind of symbiotic relationship between computer and human envisioned by Licklider in 1960. Advances in programming technology, as well as better hardware, will be required to achieve the kind of system he described. Distributed personal computing systems will help bring about these advances by providing more productive and efficient computing environments for developers.

## References

[1] N. Abramson. The ALOHA System. In *Proc. AFIPS FJCC*, pages 281–285, 1970.

[2] D.G. Bobrow et al. Tenex: A paged time-sharing system for the PDP-10. *Communications of the ACM*, 15(3):135–143, March 1972.

[3] D.R. Boggs et al. Pup: An internetwork architecture. *IEEE Trans. Comm.*, 28(4):612–624, April 1980.

[4] D.K. Brotz. *Laurel Manual.* Technical Report CSL-81-6, Xerox Palo Alto Research Center, 1981.

[5] S. Card et al. Evaluation of mouse, rate-controlled isometric joystick, step keys and text keys for text selection on a CRT. *Ergonomics*, 21(8):601–613, August 1978.

[6] Douglas W. Clark, B.W. Lampson, and Kenneth A. Pier. The memory system of a high-performance personal computer. In *The Dorado: A High-Performance Personal Computer—Three Papers, CSL-81-1*, pages 51–80, Xerox Palo Alto Research Center, 1981.

[7] L.P. Deutsch. Experience with a microprogrammed Interlisp system. *IEEE Transactions on Computers*, C-28(10), October 1979.

[8] L.P. Deutsch. A Lisp machine with very compact programs. In *Proc. 3rd IJCAI*, Stanford, 1973.

[9] L.P. Deutsch and E.A. Taft. *Requirements for an experimental programming environment.* Technical Report CSL-80-10, Xerox Palo Alto Research Center, June 1980.

[10] D.C. Engelbart. The augmented knowledge workshop. In *Proc. ACM Conf. on History of Personal Workstations*, January 1986.

[11] D.C. Engelbart. A conceptual framework for the augmentation of man's intellect. In Howerton and Weeks, editors, *Vistas in Information Handling, volume 1*, pages 1–29, Spartan Books, Washington, 1963.

[12] D.C. Engelbart and W.K English. A research center for augmenting human intellect. In *Proc. AFIPS Conf.*, pages 395–410, 1968.

[13] E.R. Fiala. The MAXC systems. *IEEE Computer*, 11(5):57–67, May 1978.

[14] J.W. Forgie. The Lincoln TX-2 input-output system. In *Proc. Western Joint Computer Conf.*, pages 156–160, February 1957.

[15] C.M. Geschke et al. Early experience with Mesa. *Communications of the ACM*, 20(8):540–553, August 1977.

[16] A. Goldberg and D. Robson. *Smalltalk-80: The Language and its Implementation.* Addison-Wesley, 1983.

[17] D. Ingalls. The Smalltalk graphics kernel. *Byte*, 6(8):168–194, August 1981.

[18] D.H. Ingalls. The Smalltalk-76 programming system: Design and implementation. In *Proc. 5th ACM Symp. Principles of Prog. Lang.*, pages 9–16, January 1978.

[19] R.K. Johnsson and J.D. Wick. An overview of the Mesa processor architecture. *ACM Sigplan Notices*, 17(4):20–29, April 1982.

[20] A.C. Kay. Microelectronics and the personal computer. *Scientific American*, 237(3):236–245, September 1977.

[21] A.C. Kay. *The Reactive Engine.* PhD thesis, University of Utah, 1969.

[22] B.W. Lampson, editor. *Alto User's Handbook.* Xerox Palo Alto Research Center, 1976.

[23] B.W. Lampson. Guest editorial. *Software-Practice and Experience*, 2:195–196, 1972.

[24] B.W. Lampson. Personal distributed computing: The Alto and Ethernet software. In *ACM Conf History of Personal Workstations*, January 1986.

[25] B.W. Lampson et al. A user machine in a time-sharing system. *Proc. IEEE*, 54(12):1744–1766, December 1966.

[26] B.W. Lampson, Gene A. McDaniel, and Severo M. Ornstein. An instruction fetch unit for a high-performance personal computer. In *The Dorado: A High-Performance Personal Computer—Three Papers, CSL-81-1*, pages 21–50, Xerox Palo Alto Research Center, 1981.

[27] B.W. Lampson and K.A. Pier. A processor for a high-performance personal computer. In *Proc. 7th Symp. Computer Arch.*, pages 146–160, ACM Sigarch/IEEE, May 1980.

[28] B.W. Lampson and R.F. Sproull. An open operating system for a single-user machine. *ACM Operating Sys. Rev.*, 13(5), November 1979.

[29] J. Licklider. Man-computer symbiosis. *IRE Trans. Human Factors in Electronics*, HFE-1:4–11, March 1960.

[30] R.M. Metcalfe and D.R. Boggs. Ethernet: Distributed packet switching for local computer networks. *Communications of the ACM*, 19(7):395–404, July 1976.

[31] R.M. Metcalfe, D.R. Boggs, C.P. Thacker, and B.W. Lampson. U.S. Patent 4,063,220: Multipoint Data Communication System With Collision Detection. December 1977.

[32] M. Richards. BCPL: A tool for compiler writing and system programming. In *AFIPS Conf. Proc.*, pages 557–566, 1969.

[33] J.F. Shoch and J.A. Hupp. Measured performance of an Ethernet local network. *Communications of the ACM*, 23(12):711–721, December 1980.

[34] W. Teitelman. A tour through Cedar. *IEEE Software*, 1(4), April 1984.

[35] C.P. Thacker. SIL—a simple illustrator for cad. In S. Chang, editor, *Fundamentals Handbook of Electrical Computer Engineering, Volume 3*, pages 477–489, Wiley, 1983.

[36] C.P. Thacker et al. Alto: A personal computer. In Siewiorek et al., editors, *Computer Structures: Principles and Examples*, chapter 33, McGraw-Hill, 1982. Also CSL-79-11, Xerox Palo Alto Research Center (1979).