7.9 Historical Perspective and Further Reading

. . . the one single development that put computers on their feet was the invention of a reliable form of memory, namely, the core memory. . . . Its cost was reasonable, it was reliable and, because it was reliable, it could in due course be made large.

Maurice Wilkes, *Memoirs of a Computer Pioneer,* 1985

This history section gives an overview of memory technologies, from mercury delay lines to DRAM, the invention of the memory hierarchy and protection mechanisms, and concludes with a brief history of operating systems, including CTSS, MULTICS, UNIX, BSD UNIX, MS-DOS, Windows, and Linux.

The developments of most of the concepts in this chapter have been driven by revolutionary advances in the technology we use for memory. Before we discuss how memory hierarchies were developed, let's take a brief tour of the development of memory technology. In this section, we focus on the technologies for building main memory and caches; Chapter 8 will provide some of the history of developments in disk technology.

The ENIAC had only a small number of registers (about 20) for its storage and implemented these with the same basic vacuum tube technology that it used for building logic circuitry. However, the vacuum tube technology was far too expensive to be used to build a larger memory capacity. Eckert came up with the idea of developing a new technology based on mercury delay lines. In this technology, electrical signals were converted into vibrations that were sent down a tube of mercury, reaching the other end, where they were read out and recirculated. One mercury delay line could store about 0.5 Kbits. Although these bits were accessed serially, the mercury delay line was about a hundred times more cost-effective than vacuum tube memory. The first known working mercury delay lines were developed at Cambridge for the EDSAC. Figure 7.9.1 shows the mercury delay lines of the EDSAC, which had 32 tanks and 512 36-bit words.

Despite the tremendous advance offered by the mercury delay lines, they were terribly unreliable and still rather expensive. The breakthrough came with the invention of core memory by J. Forrester at MIT as part of the Whirlwind project, in the early 1950s (see Figure 7.9.2). Core memory uses a ferrite core, which can be magnetized, and once magnetized, acts as a store (just as a magnetic recording tape stores information). A set of wires running through the center of the core, which had a dimension of 0.1–1.0 millimeters, make it possible to read the value stored on any ferrite core. The Whirlwind eventually included a core memory with 2048 16-bit words, or 32 Kbits. Core memory was a tremendous advance: It was cheaper, faster, much more reliable, and had higher density. Core memory was so much better than the alternatives that it became the dominant memory technology only a few years after its invention and remained so for nearly 20 years.

The technology that replaced core memory was the same one that we now use both for logic and for memory: the integrated circuit. While registers were built

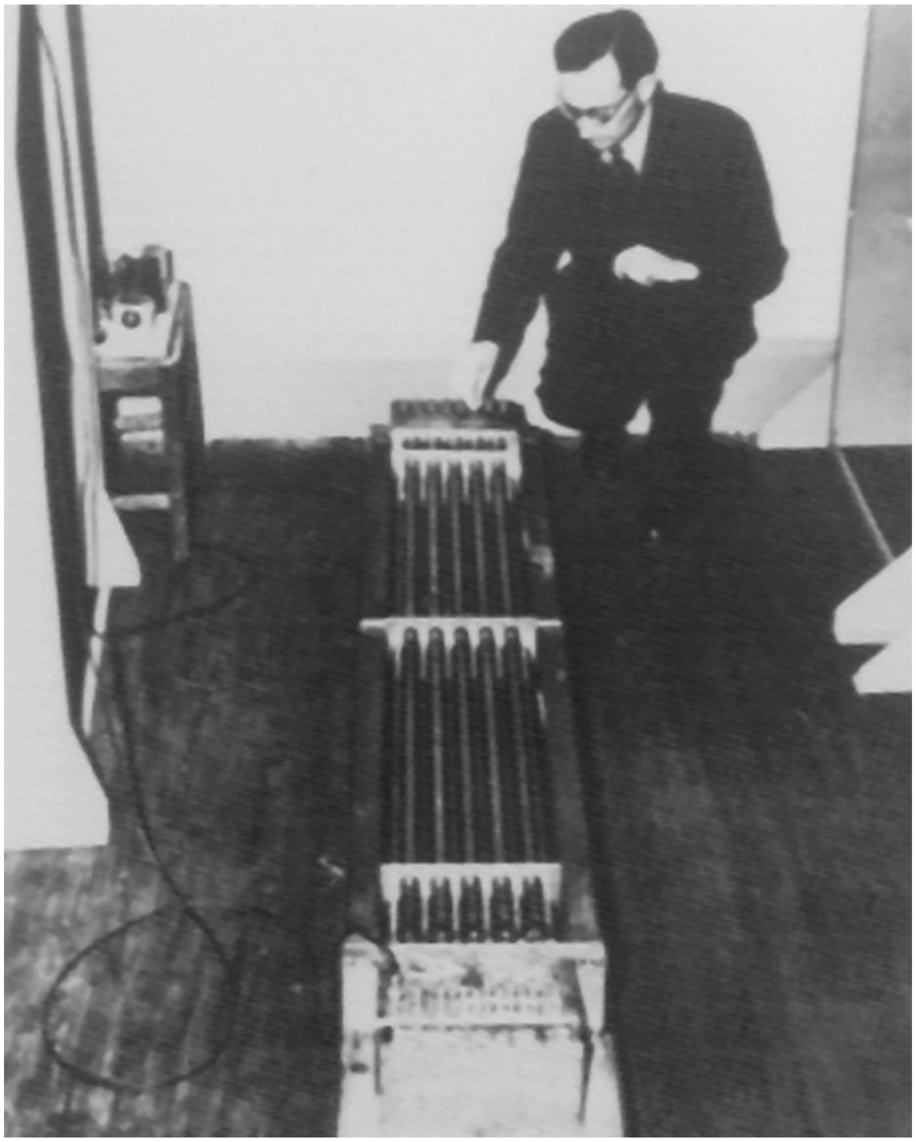

FIGURE 7.9.1 The mercury delay lines in the EDSAC. This technology made it possible to build the first stored-program computer. The young engineer in this photograph is none other than Maurice Wilkes, the lead architect of the EDSAC.

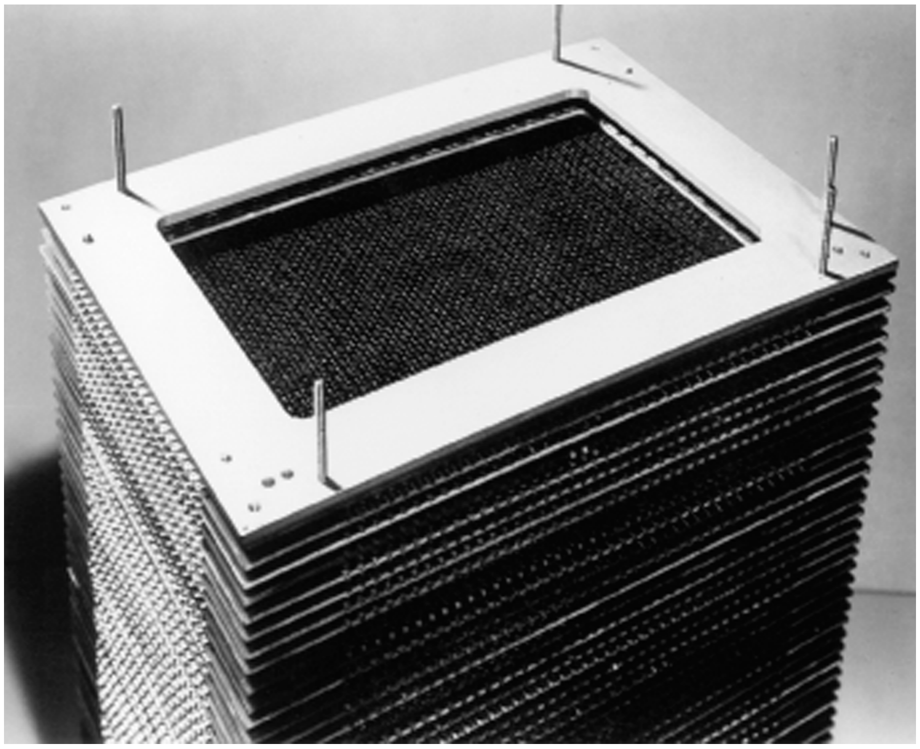

FIGURE 7.9.2 A core memory plane from the Whirlwind containing 256 cores arranged in a 16 x 16 array. Core memory was invented for the Whirlwind, which was used for air defense problems, and is now on display at the Smithsonian. (Incidentally, Ken Olsen, the founder and president of Digital for 20 years, built the computer that tested these core memories; it was his first computer.)

out of transistorized memory in the 1960s, and IBM computers used transistorized memory for microcode store and caches in 1970, building main memory out of transistors remained prohibitively expensive until the development of the integrated circuit. With the integrated circuit, it became possible to build a DRAM (dynamic random access memory—see Appendix B for a description). The first DRAMs were built at Intel in 1970, and the computers using DRAM memories (as a high-speed option to core) came shortly thereafter; they used 1 Kbit DRAMs. In fact, computer folklore says that Intel developed the microprocessor partly to help sell more DRAM. Figure 7.9.3 shows an early DRAM board. By the late 1970s, core memory became a historical curiosity. Just as core memory tech-

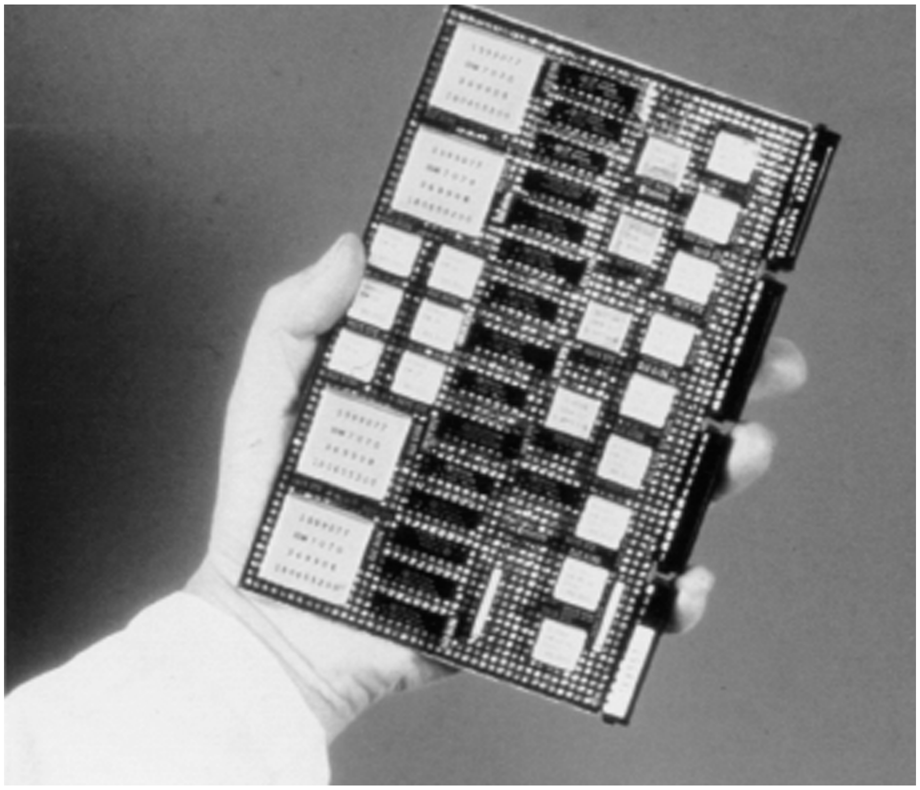

FIGURE 7.9.3 An early DRAM board. This board uses 18 Kbit chips.

nology had allowed a tremendous expansion in memory size, DRAM technology allowed a comparable expansion. In the 1990s, many personal computers have as much memory as the largest computers using core memory ever had.

Nowadays, DRAMs are typically packaged with multiple chips on a little board called a DIMM (dual in-line memory module). The SIMM (single in-line memory module) shown in Figure 7.9.4 contains a total of 1 MB and sold for about $5 in 1997. In 2004, DIMMs are available with up to 1024 MB and sell for about $100. While DRAMs will remain the dominant memory technology for some time to come, innovations in the packaging of DRAMs to provide both higher bandwidth and greater density are ongoing.

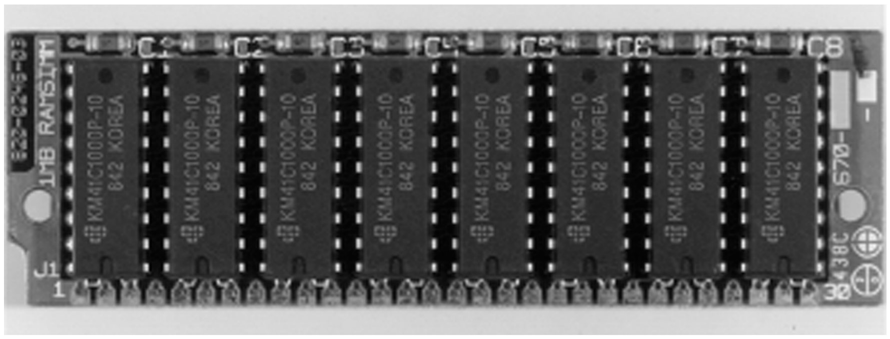

FIGURE 7.9.4 A 1 MB SIMM, built in 1986, using 1 Mbit chips. This SIMM sold for about $5/MB in 1997. In 2004 most main memory is packed in DIMMs similar to this, though using much higher-density memory chips (256 Mbit or 512 Mbit).

The Development of Memory Hierarchies

Although the pioneers of computing foresaw the need for a memory hierarchy and coined the term, the automatic management of two levels was first proposed by Kilburn and his colleagues and demonstrated at the University of Manchester with the Atlas computer, which implemented virtual memory. This was the year *before* the IBM 360 was announced. IBM planned to include virtual memory with the next generation (System/370), but the OS/360 operating system wasn't up to the challenge in 1970. Virtual memory was announced for the 370 family in 1972, and it was for this computer that the term *translation-lookaside buffer* was coined. The only computers today without virtual memory are a few supercomputers, and even they may add this feature in the near future.

The problems of inadequate address space have plagued designers repeatedly. The architects of the PDP-11 identified a small address space as the only architectural mistake from which it is difficult to recover. When the PDP-11 was designed, core memory densities were increasing at a very slow rate, and the competition from 100 other minicomputer companies meant that DEC might not have a cost-competitive product if every address had to go through the 16-bit datapath twice—hence, the decision to add just 4 more address bits than the predecessor of the PDP-11, from 16 from 12. The architects of the IBM 360 were aware of the importance of address size and planned for the architecture to extend to 32 bits of address. Only 24 bits were used in the IBM 360, however, because the low-end 360 models would have been even slower with the larger addresses. Unfortunately, the expansion effort was greatly complicated by programmers who stored extra information in the upper 8 "unused" address bits. The wider address lasted until 2000, when IBM expanded the architecture to 64 bits in the z-series.

Running out of address space has often been the cause of death for an architecture, while other architectures have managed to make the transition to a larger address space. For example, the PDP-11, a 16-bit computer, was replaced by the 32-bit VAX. The 80386 extended the 80286 architecture from a segmented 24-bit address space to a flat 32-bit address space in 1985. In the 1990s, several RISC instruction sets made the transition from 32-bit addressing to 64-bit addressing by providing a compatible extension of their instruction sets. MIPS was the first to do so. A decade later Intel and HP announced the IA-64 in large part to provide a 64-bit address successor to the 32-bit Intel IA-32 and HP Precision architectures. Architects are currently making wagers on the success of the revolutionary IA-64 versus the evolutionary AMD64, and both the winner and loser will certainly be noted in the history of computer architecture and in the boardrooms of both corporations.

Many of the early ideas in memory hierarchies originated in England. Just a few years after the Atlas paper, Wilkes [1965] published the first paper describing the concept of a cache, calling it a "slave":

> *The use is discussed of a fast core memory of, say, 32,000 words as slave to a slower core memory of, say, one million words in such a way that in practical cases the effective access time is nearer that of the fast memory than that of the slow memory.*

This two-page paper describes a direct-mapped cache. Although this was the first publication on caches, the first implementation was probably a direct-mapped instruction cache built at the University of Cambridge by Scarrott and described at the 1965 IFIP Congress. It was based on tunnel diode memory, the fastest form of memory available at the time.

Subsequent to that publication, IBM started a project that led to the first commercial computer with a cache, the IBM 360/85. Gibson at IBM recognized that memory-accessing behavior would have a significant impact on performance. He described how to measure program behavior and cache behavior and showed that the miss rate varies between programs. Using a sample of 20 programs (each with 3 million references—an incredible number for that time), Gibson analyzed the effectiveness of caches using average memory access time as the metric. Conti, Gibson, and Pitkowsky described the resulting performance of the 360/85 in the first paper to use the term *cache* in 1968. Since this early work, it has become clear that caches are one of the most important ideas not only in computer architecture, but in software systems as well. The idea of caching has found applications in operating systems, networking systems, databases, and compilers, to name a few.

There are thousands of papers on the topic of caching, and it continues to be a popular area of research.

One of the first papers on nonblocking caches was by Kroft in 1981, who may have coined the term. He later explained that he was the first to design a computer with a cache at Control Data Corporation, and when using old concepts for new mechanisms, he hit upon the idea of allowing his two-ported cache to continue to service other accesses on a miss.

Multilevel caches were the inevitable resolution to the lack of improvement in main memory latency and the higher clock rates of microprocessors. Only those in the field for a while are surprised by the size of some second- or third-level caches, as they are larger than main memories of past machines. The other surprise is that the number of levels is continuously increasing, even on a single-chip microprocessor.

Protection Mechanisms

Architectural support for protection has varied greatly over the past 20 years. In early computers, before virtual memory, protection was very simple at best. In the 1960s, more elaborate mechanisms that supported different protection levels (called *rings*) were invented. In the late 1970s and early 1980s, very elaborate mechanisms for protection were devised and later built; these mechanisms supported a variety of powerful protection schemes that allowed controlled instances of sharing, in such a way that a process could share data while controlling exactly what was done to the data. The most powerful method, called *capabilities*, created a data object that described the access rights to some portion of memory. These capabilities could then be passed to other processes, thus granting access to the object described by the capability. Supporting this sophisticated protection mechanism was both complex and costly because creation, copying, and manipulation of capabilities required a combination of operating system and hardware support. Recent computers all support a simpler protection scheme based on virtual memory, similar to that discussed in Section 7.4. Given current concerns about computer security due to the costs of worms and viruses, perhaps we will see a renaissance in protection research, potentially renewing interest in 20-year-old publications.

A Brief History of Modern Operating Systems

MIT developed the first time-sharing system, CTSS (Compatible Time-Sharing System), in 1961. John McCarthy is generally given credit for the idea of time-sharing, but Fernando Corbato was the systems person who built it. CTSS allowed three people to share a machine, and its response time of minutes or seconds was a dramatic improvement over the batch processing system it replaced. Moreover, it demonstrated the value of interactive computing.

Flush with the success of their first system, this group launched into their second system, MULTICS (Multiplexed Information and Computing Service). They included many innovations, such as strong protection, controlled sharing, and dynamic libraries. However, it suffered from the "second system effect." Fred Brooks, Jr. described the second system effect in his classic book about lessons learned from developing an operating system for the IBM mainframe, *The Mythical Man Month:*

When one is designing the successor to a relatively small, elegant, and successful system, there is a tendency to become grandiose in one's success and design an elephantine feature-laden monstrosity.

MULTICS took sharing to a logical extreme to discover the issues, including that it was too extreme. MIT, General Electric, and later Bell Labs all tried to build an economical and useful system. Despite a great deal of time and money, they did not succeed.

Berkeley was building their own time-sharing system, Cal TSS. The people leading that project included Peter Deutsch, Butler Lampson, Chuck Thacker, and Ken Thompson. They added paging virtual memory hardware to an SDS 920 and wrote an operating system for it. SDS sold this computer as the SDS-930, and it was the first commercially available time-sharing system to have operational hardware and software. Thompson graduated and joined Bell Labs. The others founded Berkeley Computer Corporation (BCC), with the goal of selling time-sharing hardware and software. We'll pick up BCC later in the story, but for now let's follow Thompson.

At Bell Labs in 1971 Thompson led the development of a simple time-sharing system that had some of the good ideas of MULTICS but left out many of the complex features. To demonstrate the contrast, it was first called UNICS. As they were joined by others at Bell Labs who had been burned from the MULTICS experience, it was renamed UNIX, with the *x* coming from Phoenix, the legendary bird that rose from the ashes.

Their result was the most elegant operating system ever built. Forced to live in the 16-bit address space of the DEC minicomputers, it had an amazing amount of functionality per line of code. Major contributions were pipes, a uniform file system, a uniform process model, and the shell user interface that allowed users to connect programs together using pipes and files.

Dennis Ritchie joined the UNIX team in 1973 from MIT, where he had experience in MULTICS, which was written in a high-level language. Like prior operating systems, UNIX had been written in assembly language. Ritchie designed a language for system implementation called C, and it was used to make UNIX portable.

Between 1971 and 1976, Bell released six editions of the UNIX time-sharing system. Thompson took a sabbatical at his alma mater and brought UNIX with him. Berkeley and many other universities began to use UNIX on the popular PDP-11 minicomputer.

When DEC announced the VAX, a 32-bit virtual address successor to the PDP-11, the question arose as what operating system should be run. UNIX became the first operating system to port to a different computer when it was ported to the VAX.

Students at Berkeley had one of the first VAXes, and they were soon adding features to UNIX for the VAX, such as paging and an very efficient implementation of the TCP/IP protocol (see Chapter 8). The Berkeley implementation of TCP/IP was notable not just because it was fast. It was essentially the *only* implementation of TCP/IP for years, since early implementations in most other operating systems consisted of copying the Berkeley code verbatim with minimal changes to integrate into the local system.

The Advanced Research Project Agency (ARPA), which funded computer science research, asked a Stanford professor, Forrest Basket, to recommend what the academic community should use: the DEC operating system VMS, led by David Cutler, or the Berkeley version of UNIX, led by a graduate student named Bill Joy? He recommended the latter, and Berkeley UNIX soon became the academic standard bearer.

The Berkeley Software Distribution (BSD) of UNIX, first released in 1978, was essentially one of the first open source movements. The sources were shipped with the tapes, and systems developers around the world learned their craft by studying the UNIX code.

BSD was also the first split of UNIX, because AT&T Bell Labs continued to develop UNIX on its own. This eventually led to a forest of UNIXes, as each company compiled the UNIX source code for their architecture. Bill Joy graduated from Berkeley and helped found Sun Microsystems, so naturally Sun OS was based on BSD UNIX. Among the many UNIX flavors was Santa Cruz Operation UNIX, HP-UX, and IBM's AIX. AT&T and Sun attempted to unify UNIX by striking a deal whereby AT&T and Sun would combine forces and jointly develop AT&T UNIX. This led to an adverse reaction from HP, IBM, and others because they did not want a competitor supplying their code, so they created the Open Source Foundation as a competing organization.

In addition to the UNIX variants from companies, public domain versions also proliferated. The BSD team at Berkeley rewrote substantial portions of UNIX so that they could distribute it without needing a license from AT&T. This eventually led to a lawsuit, which Berkeley won. BSD UNIX soon split into FreeBSD, NetBSD, and OpenBSD, provided by competing camps of developers. Apple's current operating system, OS X, is based on Free BSD.

Let’s go back to Berkeley Computer Corporation. Alas, this effort was not commercially viable. About the same time as BCC was in trouble, Xerox hired Robert Taylor to build the computer science division of the new Xerox Palo Alto Research Center (PARC) in 1970. He had just returned from a tour of duty at ARPA, where he had funded the Berkeley research. He recruited Deutsch, Lampson, and Thacker from BCC to form the core of PARC’s team: 11 of the first 20 employees were from BCC, they decided to build small computers for individuals rather than large computers for groups. This first personal computer, called the Alto, was built from the same technology as minicomputers, but it had a keyboard, mouse, graphical display, and windows. It popularized windows and led to many inventions, including client-server computing, the Ethernet, and print servers. It directly inspired the Macintosh, which was the successor to the popular Apple II.

IBM had long been interested in selling to the home, so the success of the Apple II led IBM to start a competing project. In contrast to its tradition, for this project IBM designed everything from components outside of the company. They selected the new 16-bit microprocessor from Intel, the 8086. (To lower costs, they started with the version with the 8-bit bus, called the 8088.) They visited Microsoft to see if this small company would be willing to sell their popular Basic interpreter and asked for recommendations for an operating system. Gates volunteered that Microsoft could deliver both an interpreter and an operating system, as long as they were paid a royalty fee of between $10 and $50 for each copy rather than a flat fee. IBM agreed, provided Microsoft could meet their deadlines. Microsoft didn’t have an operating system nor the time and resources to build one, but Gates knew that a Seattle company had developed an operating system for the Intel 8086. Microsoft purchased QDOS (Quick and Dirty Operating System) for $15,000, made a small change and relabeled it MS-DOS. MS-DOS was a simple operating system without any modern features—no protection, no processes, and no virtual memory—in part because they believed it wasn’t necessary for a personal computer.

Announced in 1980, the IBM PC became a tremendous success for IBM and the companies it relied upon. Microsoft sold 500,000 copies of MS-DOS by 1983, and the $10 million income allowed Microsoft to start new software projects.

After seeing a version of the Macintosh under development, Microsoft hired some people from PARC to lead its reply. The Macintosh was announced in 1984, and Windows was available on PCs the following year. It was originally an application that ran on top of DOS, but was later integrated with DOS and renamed Windows 2.0. Microsoft hired Cutler from DEC to lead the development of Windows NT, a new operating system. NT was a modern operating system with protection, processors, and so on and has much in common with DEC’s VMS. Today's PC operating systems are more sophisticated than any of the time-sharing systems of 20 years ago, yet they still suffer from the need to maintain compatibility with the crippled first PC operating systems such as MS-DOS.

The popularity of the PC led to a desire for a UNIX that ran on it. Many tried, but the most successful was written from scratch in 1991 by Linus Torvald. In addition to making the source code available like BSD, he allowed everyone to make changes and submit them for inclusion in his next release. Linux popularized open source development as we know it today, with such software getting hundreds of volunteers to test releases and add new features.

Many people in this story won awards for their roles in the development of modern operating systems. McCarthy received an ACM Turing Award in 1971 in part for his contributions to time-sharing. In 1983 Thompson and Ritchie received it for UNIX. The announcement said that "the genius of the UNIX system is its framework, which enables programmers to stand on the work of others." In 1990, Corbato received the Turing Award for his contributions to CTSS and MULTICS. Two years later, Lampson won it in part for his work on personal computing and operating systems.

Further Reading

Cantin, J. F., and M. D. Hill [2001]. "Cache performance for selected SPEC CPU2000 benchmarks," *SIGARCH Computer Architecture News*, 29:4 (September), 13–18.

A reference paper of cache miss rates for many cache sizes for the SPEC2000 benchmarks.

Conti, C., D. H. Gibson, and S. H. Pitowsky [1968]. "Structural aspects of the System/360 Model 85, part I: General organization," *IBM Systems J.* 7:1, 2–14.

A classic paper that describes the first commercial computer to use a cache and its resulting performance.

Hennessy, J., and D. Patterson [2003]. Chapter 5 in *Computer Architecture: A Quantitative Approach*, third ed., San Francisco: Morgan Kaufmann Publishers.

For more in-depth coverage of a variety of topics including protection, cache performance of out-of-order processors, virtually addressed caches, multilevel caches, compiler optimizations, additional latency tolerance mechanisms, and cache coherency.

Kilburn, T., D. B. G. Edwards, M. J. Lanigan, and F. H. Sumner [1962]. "One-level storage system," *IRE Transactions on Electronic Computers* EC-11 (April) 223–35. Also appears in D. P. Siewiorek, C. G. Bell, and A. Newell [1982] *Computer Structures: Principles and Examples*, New York: McGraw-Hill, 135–48.

This classic paper is the first proposal for virtual memory.

LaMarca, A., and R. E. Ladner [1996]. "The influence of caches on the performance of heaps," *ACM J. of Experimental Algorithmics,* vol. 1.

This paper shows the difference between complexity analysis of an algorithm, instruction count performance, and memory hierarchy for four sorting algorithms.

Przybylski, S. A. [1990]. *Cache and Memory Hierarchy Design: A Performance-Directed Approach*, San Francisco: Morgan Kaufmann Publishers.

A thorough exploration of multilevel memory hierarchies and their performance.

Ritchie, D. [1984]. "The evolution of the UNIX time-sharing system." *AT& T Bell Laboratories Technical Journal*, 1984, 1577–93.

The history of UNIX from one of its inventors.

Ritchie, D. M., and K. Thompson [1978]. The UNIX time-sharing system," *Bell System Technical Journal*, August, 1991–2019.

A paper describing the most elegant operating system ever invented.

Silberschatz, A., P. Galvin, and G. Grange [2003]. *Operating System Concepts*, sixth ed., Reading, MA: Addison-Wesley.

An operating systems textbook with a thorough discussion of virtual memory, processes and process management, and protection issues.

Smith, A. J. [1982]. "Cache memories," *Computing Surveys* 14:3 (September), 473–530.

The classic survey paper on caches. This paper defined the terminology for the field and has served as a reference for many computer designers.

Smith, D. K., and R.C. Alexander. [1988]. *Fumbling the Future: How Xerox Invented, Then Ignored, the First Personal Computer*, New York: Morrow.

A popular book that explains the role of Xerox PARC in laying the foundation for today's computing, which Xerox did not substantially benefit from.

Tanenbaum, A. [2001]. *Modern Operating Systems*, second ed., Upper Saddle River, NJ: Prentice-Hall.

An operating system textbook with a good discussion of virtual memory.

Wilkes, M. [1965]. "Slave memories and dynamic storage allocation," *IEEE Trans. Electronic Computers* EC-14:2 (April), 270–71.

The first classic paper on caches.