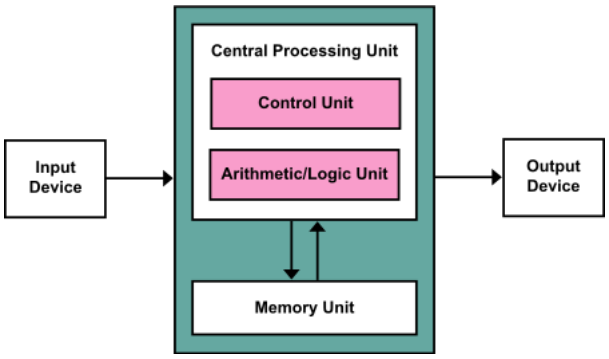
WIKIPEDIA

Von Neumann architecture

The **von Neumann architecture** — also known as the **von Neumann model** or **Princeton architecture** — is a computer architecture based on a 1945 description by John von Neumann, and by others, in the *First Draft of a Report on the EDVAC*.[1] The document describes a design architecture for an electronic digital computer with these components:

- A processing unit with both an arithmetic logic unit and processor registers
- A control unit that includes an instruction register and a program counter
- Memory that stores data and instructions
- External mass storage
- Input and output mechanisms[1][2]



A von Neumann architecture scheme

The term "von Neumann architecture" has evolved to refer to any stored-program computer in which an instruction fetch and a data operation cannot occur at the same time (since they share a common bus). This is referred to as the von Neumann bottleneck, which often limits the performance of the corresponding system.[3]

The design of a von Neumann architecture machine is simpler than in a Harvard architecture machine—which is also a stored-program system, yet has one dedicated set of address and data buses for reading and writing to memory, and another set of address and data buses to fetch instructions.

A stored-program digital computer keeps both program instructions and data in read–write, random-access memory (RAM). Stored-program computers were an advancement over the program-controlled computers of the 1940s, such as the Colossus and the ENIAC. Those were programmed by setting switches and inserting patch cables to route data and control signals between various functional units. The vast majority of modern computers use the same memory for both data and program instructions, but have caches between the CPU and memory, and, for the caches closest to the CPU, have separate caches for instructions and data, so that most instruction and data fetches use separate buses (split cache architecture).

Contents

History

The earliest computing machines had fixed programs. Some very simple computers still use this design, either for simplicity or training purposes. For example, a desk calculator (in principle) is a fixed program computer. It can do basic mathematics, but it cannot run a word processor or games. Changing the program of a fixed-program machine requires rewiring, restructuring, or redesigning the machine. The earliest computers were not so much "programmed" as "designed" for a particular task. "Reprogramming" – when possible at all – was a laborious process that started with flowcharts and paper notes, followed by detailed engineering designs, and then the often-arduous process of physically rewiring and rebuilding the machine. It could take three weeks to set up and debug a program on ENIAC.[4]

With the proposal of the stored-program computer, this changed. A stored-program computer includes, by design, an instruction set, and can store in memory a set of instructions (a program) that details the computation.

A stored-program design also allows for self-modifying code. One early motivation for such a facility was the need for a program to increment or otherwise modify the address portion of instructions, which operators had to do manually in early designs. This became less important when index registers and indirect addressing became usual features of machine architecture. Another use was to embed frequently used data in the instruction stream using immediate addressing. Self-modifying code has largely fallen out of favor, since it is usually hard to understand and debug, as well as being inefficient under modern processor pipelining and caching schemes.

Capabilities

On a large scale, the ability to treat instructions as data is what makes assemblers, compilers, linkers, loaders, and other automated programming tools possible. It makes "programs that write programs" possible.[5] This has made a sophisticated self-hosting computing ecosystem flourish around von Neumann architecture machines.

Some high level languages leverage the von Neumann architecture by providing an abstract, machine-independent way to manipulate executable code at runtime (e.g., LISP), or by using runtime information to tune just-in-time compilation (e.g. languages hosted on the Java virtual machine, or languages embedded in web browsers).

On a smaller scale, some repetitive operations such as BITBLT or pixel and vertex shaders can be accelerated on general purpose processors with just-in-time compilation techniques. This is one use of self-modifying code that has remained popular.

Development of the stored-program concept

The mathematician Alan Turing, who had been alerted to a problem of mathematical logic by the lectures of Max Newman at the University of Cambridge, wrote a paper in 1936 entitled *On Computable Numbers, with an Application to the Entscheidungsproblem*, which was published in the *Proceedings of the London Mathematical Society*.[6] In it he described a hypothetical machine he called a *universal computing machine*, now known as the "Universal Turing machine". The hypothetical machine had an infinite store (memory in today's terminology) that contained both instructions and data. John von Neumann became acquainted with Turing while he was a visiting professor at Cambridge in 1935, and also during Turing's PhD year at the Institute for Advanced Study in Princeton, New Jersey during 1936–1937. Whether he knew of Turing's paper of 1936 at that time is not clear.

In 1936, Konrad Zuse also anticipated, in two patent applications, that machine instructions could be stored in the same storage used for data.[7]

Independently, J. Presper Eckert and John Mauchly, who were developing the ENIAC at the Moore School of Electrical Engineering of the University of Pennsylvania, wrote about the stored-program concept in December 1943. [8][9] In planning a new machine, EDVAC, Eckert wrote in January 1944 that they would store data and programs in a new addressable memory device, a mercury metal delay-line memory. This was the first time the construction of a practical stored-program machine was proposed. At that time, he and Mauchly were not aware of Turing's work.

Von Neumann was involved in the Manhattan Project at the Los Alamos National Laboratory. It required huge amounts of calculation, and thus drew him to the ENIAC project, during the summer of 1944. There he joined the ongoing discussions on the design of this stored-program computer, the EDVAC. As part of that group, he wrote up a description titled *First Draft of a Report on the EDVAC*[1] based on the work of Eckert and Mauchly. It was unfinished when his colleague Herman Goldstine circulated it, and bore only von Neumann's name (to the consternation of Eckert and Mauchly).[10] The paper was read by dozens of von Neumann's colleagues in America and Europe, and influenced the next round of computer designs.

Jack Copeland considers that it is "historically inappropriate to refer to electronic stored-program digital computers as 'von Neumann machines' ".[11] His Los Alamos colleague Stan Frankel said of von Neumann's regard for Turing's ideas[12]

> I know that in or about 1943 or '44 von Neumann was well aware of the fundamental importance of Turing's paper of 1936…. Von Neumann introduced me to that paper and at his urging I studied it with care. Many people have acclaimed von Neumann as the "father of the computer" (in a modern sense of the term) but I am sure that he would never have made that mistake himself. He might well be called the midwife, perhaps, but he firmly emphasized to me, and to others I am sure, that the fundamental conception is owing to Turing— in so far as not anticipated by Babbage…. Both Turing and von Neumann, of course, also made substantial contributions to the "reduction to practice" of these concepts

> but I would not regard these as comparable in importance with the introduction and explication of the concept of a computer able to store in its memory its program of activities and of modifying that program in the course of these activities.

At the time that the "First Draft" report was circulated, Turing was producing a report entitled *Proposed Electronic Calculator*. It described in engineering and programming detail, his idea of a machine he called the *Automatic Computing Engine (ACE)*.[13] He presented this to the executive committee of the British National Physical Laboratory on February 19, 1946. Although Turing knew from his wartime experience at Bletchley Park that what he proposed was feasible, the secrecy surrounding Colossus, that was subsequently maintained for several decades, prevented him from saying so. Various successful implementations of the ACE design were produced.

Both von Neumann's and Turing's papers described stored-program computers, but von Neumann's earlier paper achieved greater circulation and the computer architecture it outlined became known as the "von Neumann architecture". In the 1953 publication *Faster than Thought: A Symposium on Digital Computing Machines* (edited by B. V. Bowden), a section in the chapter on *Computers in America* reads as follows:[14]

> **The Machine of the Institute For Advanced Studies, Princeton**
>
> In 1945, Professor J. von Neumann, who was then working at the Moore School of Engineering in Philadelphia, where the E.N.I.A.C. had been built, issued on behalf of a group of his co-workers, a report on the logical design of digital computers. The report contained a detailed proposal for the design of the machine that has since become known as the E.D.V.A.C. (electronic discrete variable automatic computer). This machine has only recently been completed in America, but the von Neumann report inspired the construction of the E.D.S.A.C. (electronic delay-storage automatic calculator) in Cambridge (see page 130).
>
> In 1947, Burks, Goldstine and von Neumann published another report that outlined the design of another type of machine (a parallel machine this time) that would be exceedingly fast, capable perhaps of 20,000 operations per second. They pointed out that the outstanding problem in constructing such a machine was the development of suitable memory with instantaneously accessible contents. At first they suggested using a special vacuum tube—called the "Selectron"—which the Princeton Laboratories of RCA had invented. These tubes were expensive and difficult to make, so von Neumann subsequently decided to build a machine based on the Williams memory. This machine—completed in June, 1952 in Princeton—has become popularly known as the Maniac. The design of this machine inspired at least half a dozen machines now being built in America, all known affectionately as "Johniacs".

In the same book, the first two paragraphs of a chapter on ACE read as follows:[15]

> **Automatic Computation at the National Physical Laboratory**
>
> One of the most modern digital computers which embodies developments and improvements in the technique of automatic electronic computing was recently demonstrated at the National Physical Laboratory, Teddington, where it has been

> designed and built by a small team of mathematicians and electronics research engineers on the staff of the Laboratory, assisted by a number of production engineers from the English Electric Company, Limited. The equipment so far erected at the Laboratory is only the pilot model of a much larger installation which will be known as the Automatic Computing Engine, but although comparatively small in bulk and containing only about 800 thermionic valves, as can be judged from Plates XII, XIII and XIV, it is an extremely rapid and versatile calculating machine.
>
> The basic concepts and abstract principles of computation by a machine were formulated by Dr. A. M. Turing, F.R.S., in a paper[1]. read before the London Mathematical Society in 1936, but work on such machines in Britain was delayed by the war. In 1945, however, an examination of the problems was made at the National Physical Laboratory by Mr. J. R. Womersley, then superintendent of the Mathematics Division of the Laboratory. He was joined by Dr. Turing and a small staff of specialists, and, by 1947, the preliminary planning was sufficiently advanced to warrant the establishment of the special group already mentioned. In April, 1948, the latter became the Electronics Section of the Laboratory, under the charge of Mr. F. M. Colebrook.

Early von Neumann-architecture computers

The *First Draft* described a design that was used by many universities and corporations to construct their computers.[16] Among these various computers, only ILLIAC and ORDVAC had compatible instruction sets.

- ARC2 (Birkbeck, University of London) officially came online on May 12, 1948.[17]
- Manchester Baby (Victoria University of Manchester, England) made its first successful run of a stored program on June 21, 1948.
- EDSAC (University of Cambridge, England) was the first practical stored-program electronic computer (May 1949)
- Manchester Mark 1 (University of Manchester, England) Developed from the Baby (June 1949)
- CSIRAC (Council for Scientific and Industrial Research) Australia (November 1949)
- MESM in Kyiv, Ukraine (November 1950)
- EDVAC (Ballistic Research Laboratory, Computing Laboratory at Aberdeen Proving Ground 1951)
- ORDVAC (U-Illinois) at Aberdeen Proving Ground, Maryland (completed November 1951)[18]
- IAS machine at Princeton University (January 1952)
- MANIAC I at Los Alamos Scientific Laboratory (March 1952)
- ILLIAC at the University of Illinois, (September 1952)
- BESM-1 in Moscow (1952)
- AVIDAC at Argonne National Laboratory (1953)
- ORACLE at Oak Ridge National Laboratory (June 1953)
- BESK in Stockholm (1953)
- JOHNNIAC at RAND Corporation (January 1954)
- DASK in Denmark (1955)
- WEIZAC at the Weizmann Institute of Science in Rehovot, Israel (1955)
- PERM in Munich (1956)
- SILLIAC in Sydney (1956)

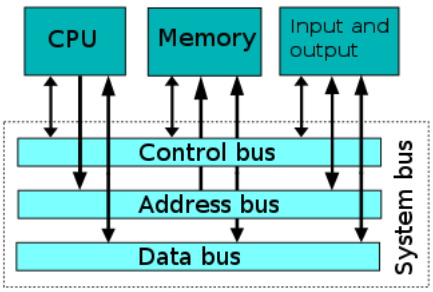
Early stored-program computers

The date information in the following chronology is difficult to put into proper order. Some dates are for first running a test program, some dates are the first time the computer was demonstrated or completed, and some dates are for the first delivery or installation.

- The IBM SSEC had the ability to treat instructions as data, and was publicly demonstrated on January 27, 1948. This ability was claimed in a US patent.[19][20] However it was partially electromechanical, not fully electronic. In practice, instructions were read from paper tape due to its limited memory.[21]
- The ARC2 developed by Andrew Booth and Kathleen Booth at Birkbeck, University of London officially came online on May 12, 1948.[17] It featured the first rotating drum storage device.[22][23]
- The Manchester Baby was the first fully electronic computer to run a stored program. It ran a factoring program for 52 minutes on June 21, 1948, after running a simple division program and a program to show that two numbers were relatively prime.
- The ENIAC was modified to run as a primitive read-only stored-program computer (using the Function Tables for program ROM) and was demonstrated as such on September 16, 1948, running a program by Adele Goldstine for von Neumann.
- The BINAC ran some test programs in February, March, and April 1949, although was not completed until September 1949.
- The Manchester Mark 1 developed from the Baby project. An intermediate version of the Mark 1 was available to run programs in April 1949, but was not completed until October 1949.
- The EDSAC ran its first program on May 6, 1949.
- The EDVAC was delivered in August 1949, but it had problems that kept it from being put into regular operation until 1951.
- The CSIR Mk I ran its first program in November 1949.
- The SEAC was demonstrated in April 1950.
- The Pilot ACE ran its first program on May 10, 1950, and was demonstrated in December 1950.
- The SWAC was completed in July 1950.
- The Whirlwind was completed in December 1950 and was in actual use in April 1951.
- The first ERA Atlas (later the commercial ERA 1101/UNIVAC 1101) was installed in December 1950.

Evolution

Through the decades of the 1960s and 1970s computers generally became both smaller and faster, which led to evolutions in their architecture. For example, memory-mapped I/O lets input and output devices be treated the same as memory.[24] A single system bus could be used to provide a modular system with lower cost. This is sometimes called a "streamlining" of the architecture.[25] In subsequent decades, simple microcontrollers would sometimes omit features of the model to lower cost and size. Larger computers added features for higher performance.



Single system bus evolution of the architecture

Design limitations

Von Neumann bottleneck

The shared bus between the program memory and data memory leads to the *von Neumann bottleneck*, the limited throughput (data transfer rate) between the central processing unit (CPU) and memory compared to the amount of memory. Because the single bus can only access one of the two classes of memory at a time, throughput is lower than the rate at which the CPU can work. This seriously limits the effective processing speed when the CPU is required to perform minimal processing on large amounts of data. The CPU is continually forced to wait for needed data to move to or from memory. Since CPU speed and memory size have increased much faster than the throughput between them, the bottleneck has become more of a problem, a problem whose severity increases with every new generation of CPU.

The von Neumann bottleneck was described by John Backus in his 1977 ACM Turing Award lecture. According to Backus:

> Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words back and forth through the von Neumann bottleneck. Not only is this tube a literal bottleneck for the data traffic of a problem, but, more importantly, it is an intellectual bottleneck that has kept us tied to word-at-a-time thinking instead of encouraging us to think in terms of the larger conceptual units of the task at hand. Thus programming is basically planning and detailing the enormous traffic of words through the von Neumann bottleneck, and much of that traffic concerns not significant data itself, but where to find it.[26][27][28]

Mitigations

There are several known methods for mitigating the Von Neumann performance bottleneck. For example, the following all can improve performance:

- Providing a cache between the CPU and the main memory
- providing separate caches or separate access paths for data and instructions (the so-called Modified Harvard architecture)
- using branch predictor algorithms and logic
- providing a limited CPU stack or other on-chip scratchpad memory to reduce memory access
- Implementing the CPU and the memory hierarchy as a system on chip, providing greater locality of reference and thus reducing latency and increasing throughput between processor registers and main memory

The problem can also be sidestepped somewhat by using parallel computing, using for example the non-uniform memory access (NUMA) architecture—this approach is commonly employed by supercomputers. It is less clear whether the *intellectual bottleneck* that Backus criticized has changed much since 1977. Backus's proposed solution has not had a major influence. Modern functional programming and object-oriented programming are much less geared towards "pushing vast numbers of words back and forth" than earlier languages like FORTRAN were, but internally, that is still what computers spend much of their time doing, even highly parallel supercomputers.

As of 1996, a database benchmark study found that three out of four CPU cycles were spent waiting for memory. Researchers expect that increasing the number of simultaneous instruction streams with multithreading or single-chip multiprocessing will make this bottleneck even worse.[29] In the context of multi-core processors, additional overhead is required to maintain cache coherence between processors and threads.

Self-modifying code

Aside from the von Neumann bottleneck, program modifications can be quite harmful, either by accident or design. In some simple stored-program computer designs, a malfunctioning program can damage itself, other programs, or the operating system, possibly leading to a computer crash. Memory protection and other forms of access control can usually protect against both accidental and malicious program changes.

See also

- CARDboard Illustrative Aid to Computation
- Interconnect bottleneck
- Little man computer
- Random-access machine
- Harvard architecture
- Turing machine
- Eckert architecture

References

1. von Neumann, John (1945), *First Draft of a Report on the EDVAC* (https://web.archive.org/web/20130314123032/http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf) (PDF), archived from the original (https://sites.google.com/site/michaeldgodfrey/vonneumann/vnedvac.pdf) (PDF) on March 14, 2013, retrieved August 24, 2011.
2. Ganesan 2009.
3. Markgraf, Joey D. (2007), *The Von Neumann Bottleneck* (https://web.archive.org/web/20131212205159/http://aws.linnbenton.edu/cs271c/markgrj/), archived from the original (http://aws.linnbenton.edu/cs271c/markgrj/) on December 12, 2013.
4. Copeland 2006, p. 104.
5. MFTL *(My Favorite Toy Language) entry Jargon File 4.4.7* (http://catb.org/~esr/jargon/html/M/MFTL.html), retrieved July 11, 2008.
6. Turing, Alan M. (1936), "On Computable Numbers, with an Application to the Entscheidungsproblem", *Proceedings of the London Mathematical Society*, 2 (published 1937), vol. 42, pp. 230–265, doi:10.1112/plms/s2-42.1.230 (https://doi.org/10.1112%2Fplms%2Fs2-42.1.230), S2CID 73712 (https://api.semanticscholar.org/CorpusID:73712) and Turing, Alan M. (1938), "On Computable Numbers, with an Application to the Entscheidungsproblem. A correction", *Proceedings of the London Mathematical Society*, 2 (published 1937), vol. 43, no. 6, pp. 544–546, doi:10.1112/plms/s2-43.6.544 (https://doi.org/10.1112%2Fplms%2Fs2-43.6.544).

7. Williams, F. C.; Kilburn, T. (September 25, 1948), "Electronic Digital Computers" (https://web.archive.org/web/20090406014626/http://www.computer50.org/kgill/mark1/natletter.html), *Nature*, **162** (4117): 487, Bibcode:1948Natur.162..487W (https://ui.adsabs.harvard.edu/abs/1948Natur.162..487W), doi:10.1038/162487a0 (https://doi.org/10.1038%2F162487a0), S2CID 4110351 (https://api.semanticscholar.org/CorpusID:4110351), archived from the original (http://www.computer50.org/kgill/mark1/natletter.html) on April 6, 2009, retrieved April 10, 2009.
8. Lukoff, Herman (1979). *From Dits to Bits: A personal history of the electronic computer*. Portland, Oregon, USA: Robotics Press. ISBN 0-89661-002-0. LCCN 79-90567 (https://lccn.loc.gov/79-90567).
9. ENIAC project administrator Grist Brainerd's December 1943 progress report for the first period of the ENIAC's development implicitly proposed the stored program concept (while simultaneously rejecting its implementation in the ENIAC) by stating that "in order to have the simplest project and not to complicate matters", the ENIAC would be constructed without any "automatic regulation".
10. Copeland 2006, p. 113.
11. Copeland, Jack (2000), *A Brief History of Computing: ENIAC and EDVAC* (http://www.alanturing.net/turing_archive/pages/Reference%20Articles/BriefHistofComp.html#ACE), retrieved January 27, 2010.
12. Copeland, Jack (2000), *A Brief History of Computing: ENIAC and EDVAC* (http://www.alanturing.net/turing_archive/pages/Reference%20Articles/BriefHistofComp.html#ACE), retrieved January 27, 2010 (a work which cites Randell, Brian (1972), Meltzer, B.; Michie, D. (eds.), "On Alan Turing and the Origins of Digital Computers", *Machine Intelligence*, Edinburgh: Edinburgh University Press, **7**: 10, ISBN 0-902383-26-4.
13. Copeland 2006, pp. 108–111.
14. Bowden 1953, pp. 176, 177.
15. Bowden 1953, p. 135.
16. "Electronic Computer Project" (http://www.ias.edu/people/vonneumann/ecp/). Institute for Advanced Study. September 11, 2009. Retrieved May 26, 2011.
17. Campbell-Kelly, Martin (April 1982). "The Development of Computer Programming in Britain (1945 to 1955)". *IEEE Annals of the History of Computing*. **4** (2): 121–139. doi:10.1109/MAHC.1982.10016 (https://doi.org/10.1109%2FMAHC.1982.10016). S2CID 14861159 (https://api.semanticscholar.org/CorpusID:14861159).
18. Robertson, James E. (1955), *Illiac Design Techniques*, report number UIUCDCS-R-1955-146, Digital Computer Laboratory, University of Illinois at Urbana-Champaign.
19. Selective Sequence Electronic Calculator (USPTO Web site) (http://patft.uspto.gov/netacgi/nph-Parser?patentnumber=2636672).
20. Selective Sequence Electronic Calculator (Google Patents) (https://patents.google.com/patent/US2636672).
21. Grosch, Herbert R. J. (1991), *Computer: Bit Slices From a Life* (http://www.columbia.edu/acis/history/computer.html), Third Millennium Books, ISBN 0-88733-085-1.
22. Lavington, Simon, ed. (2012). *Alan Turing and his Contemporaries: Building the World's First Computers*. London: British Computer Society. p. 61. ISBN 9781906124908.
23. Johnson, Roger (April 2008). "School of Computer Science & Information Systems: A Short History" (http://www.dcs.bbk.ac.uk/site/assets/files/1029/50yearsofcomputing.pdf) (PDF). *Birkbeck College*. University of London. Retrieved July 23, 2017.
24. Bell, C. Gordon; Cady, R.; McFarland, H.; O'Laughlin, J.; Noonan, R.; Wulf, W. (1970), "A New Architecture for Mini-Computers—The DEC PDP-11" (http://research.microsoft.com/en-us/um/people/gbell/CGB%20Files/New%20Architecture%20PDP11%20SJCC%201970%20c.pdf) (PDF), *Spring Joint Computer Conference*, pp. 657–675.

25. Null, Linda; Lobur, Julia (2010), *The essentials of computer organization and architecture* (https://books.google.com/books?id=f83XxoBC_8MC&pg=PA36) (3rd ed.), Jones & Bartlett Learning, pp. 36, 199–203, ISBN 978-1-4496-0006-8.
26. Backus, John W. "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs" (https://doi.org/10.1145%2F359576.359579). doi:10.1145/359576.359579 (https://doi.org/10.1145%2F359576.359579).
27. Dijkstra, Edsger W. "E. W. Dijkstra Archive: A review of the 1977 Turing Award Lecture" (http://www.cs.utexas.edu/~EWD/transcriptions/EWD06xx/EWD692.html). Retrieved July 11, 2008.
28. Backus, John (August 1978). "Can Programming Be Liberated from the von Neumann Style? A Functional Style and Its Algebra of Programs" (https://www.cs.cmu.edu/~crary/819-f09/Backus78.pdf) (PDF). *Communications of the ACM*. **21** (8): 613–641. doi:10.1145/359576.359579 (https://doi.org/10.1145%2F359576.359579). S2CID 16367522 (https://api.semanticscholar.org/CorpusID:16367522). Retrieved September 19, 2020 – via Karl Crary, School of Computer Science, Carnegie Mellon University.
29. Sites, Richard L.; Patt, Yale. "Architects Look to Processors of Future" (http://cva.stanford.edu/classes/cs99s/papers/architects_look_to_future.pdf). Microprocessor report. 1996.

Further reading

- Bowden, B. V., ed. (1953), *Faster Than Thought: A Symposium on Digital Computing Machines*, London: Sir Isaac Pitman and Sons Ltd.
- Rojas, Raúl; Hashagen, Ulf, eds. (2000), *The First Computers: History and Architectures*, MIT Press, ISBN 0-262-18197-5
- Davis, Martin (2000), *The universal computer: the road from Leibniz to Turing*, New York: W. W. Norton & Company Inc., ISBN 0-393-04785-7 republished as: Davis, Martin (2001), *Engines of Logic: Mathematicians and the Origin of the Computer*, New York: W. W. Norton & Company, ISBN 978-0-393-32229-3
- *Can Programming be Liberated from the von Neumann Style?*. Backus, John. 1977 ACM Turing Award Lecture. Communications of the ACM, August 1978, Volume 21, Number 8 Online PDF (http://www.stanford.edu/class/cs242/readings/backus.pdf) Archived (https://web.archive.org/web/20070621162552/http://www.stanford.edu/class/cs242/readings/backus.pdf) June 21, 2007, at the Wayback Machine see details at https://www.cs.tufts.edu/~nr/backus-lecture.html
- Bell, C. Gordon; Newell, Allen (1971), *Computer Structures: Readings and Examples*, McGraw-Hill Book Company, New York. Massive (668 pages)
- Copeland, Jack (2006), "Colossus and the Rise of the Modern Computer", in Copeland, B. Jack (ed.), *Colossus: The Secrets of Bletchley Park's Codebreaking Computers*, Oxford: Oxford University Press, ISBN 978-0-19-284055-4
- Ganesan, Deepak (2009), *The von Neumann Model* (https://web.archive.org/web/20120425083227/http://none.cs.umass.edu/~dganesan/courses/fall09/handouts/Chapter4.pdf) (PDF), archived from the original (http://none.cs.umass.edu/~dganesan/courses/fall09/handouts/Chapter4.pdf) (PDF) on April 25, 2012, retrieved October 22, 2011
- McCartney, Scott (1999). *ENIAC: The Triumphs and Tragedies of the World's First Computer* (https://archive.org/details/eniac00scot). Walker & Co. ISBN 0-8027-1348-3.
- Goldstine, Herman H. (1972). *The Computer from Pascal to von Neumann* (https://archive.org/details/computerfrompasc00herm). Princeton University Press. ISBN 0-691-08104-2.
- Shurkin, Joel (1984). *Engines of the Mind: A history of the Computer*. New York, London: W. W. Norton & Company. ISBN 0-393-01804-0.

External links

- Harvard vs von Neumann (http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.faqs/ka11516.html)
- A tool that emulates the behavior of a von Neumann machine (https://web.archive.org/web/20080219131555/http://home.gna.org/vov/)
- JOHNNY: A simple Open Source simulator of a von Neumann machine for educational purposes (http://sourceforge.net/projects/johnnysimulator/)