Existence Precedes Essence - Meaning of the Stored-Program Concept

Allan Olley

Existence Precedes Essence - Meaning of the Stored-Program Concept

Allan Olley

IHPST, University of Toronto
91 Charles St., Toronto, ON, Canada, allan.olley@utoronto.ca

Abstract. The emergence of electronic stored-program computers in contain the 1940s marks a break with past developments in machine calculation. Drawing on the work of various historians, I attempt to define the essence of that break and therefore of the modern computer. I conclude that the generally used distinction between computers and precursor machines in terms of the stored-program concept and von Neumann architecture rests not only on differences in hardware but also in the programming and use of machines. Next I discuss the derived definition in terms of machines from the 1940s and 50s to elucidate the definition's implications for the history of computing.

Keywords: Stored-program, von Neumann architecture, computer history, computer architecture, history of software, IBM SSEC.

1 Introduction

The July 1939 issue of Astounding Science-Fiction included an article entitled "Tools for Brains" by one Leo Vernon. This article included a brief history of calculating machines, including mention of Charles Babbage and short discussion of recent scientific uses of punched card machines at Columbia and differential analyzers at MIT. He concludes by speculating on future machines sketching what a "dream machine" for such scientific computation might look like:

> *"The dream machine may fill an entire building. It will be operated from a central control room made up entirely of switchboard panels, operated by trained mathematicians, and an automatic printer giving back the results. A physicist has spent months getting a problem ready for solution. He has long tables of numbers, experimental data. These numbers have to be combined and recombined with still more numbers, producing hundreds of thousands or millions of numbers..."* [1]

Vernon imagined the subsequent calculation occurring in minutes and with the aid of mathematical tables punched onto paper tape. [1]

Vernon's predictions seem in some aspects prescient and in some aspects completely wrong. Computing machines soon did become very large and very fast, but soon Vernon's dreams of a machine operated by a room full of switchboards would prove more a nightmare and mathematical tables would fade from use. The point of this paper is to illuminate the source of such predictions. Vernon had just

finished reviewing developments and practice in scientific computing up to 1939. In part his prediction simply projected an extension of previous practice. Punched card machines used switchboards and tables punched into machine readable forms. Therefore so would the machines of the future. The developments during and immediately after World War II changed, or added to, the nature of machine computation in fundamental ways.

Many attempts have been made to capture the essence of the modern computer. The goals implicit in such definitions have been various, including attempts to characterize a “computer age” distinct from what came before it and the less reputable task of establishing priority for inventors. A requirement given by most analysts for a modern computer is that it be electronic. Electronic speeds made computers capable of thousands of computations in a second, far outstripping the speed of any human effort. However, Vernon’s description of the dream machine implies a machine with just such prodigious speeds. So it is not speed alone that separates Vernon’s dream from later developments. In parallel most commentaries suggest that the speed of the modern computer is only part of the story. Accounts point to the design elements that make a modern computer a general purpose machine applicable to a wide range of problems.

An example of these developments in machine computation is the ENIAC. The ENIAC, Electronic Numerical Integrator and Computer, began operation in 1946 and resembles Vernon’s dream machine. The ENIAC’s electronic tubes allowed computation at speeds a thousand times faster than previous machines and operators controlled the ENIAC by rewiring switch boards. However, the physical rewiring of the ENIAC proved onerous and a new system was implemented in 1948. This system controlled the ENIAC’s operations using a series of instructions encoded in a numerical format, a program. The operators preferred the new system of set-up despite a significant reduction in the speed of actual computation. [2]

Traditionally historians attribute the articulation of key design elements of the modern computer to the “Draft report on the EDVAC.” The report became synonymous with John von Neumann, but was based on the work of a team at the Moore school, responsible for constructing the ENIAC, lead by John Mauchly and Presper Eckert. The EDVAC was to be the successor to the ENIAC and an improvement upon it. The report details various broad design features of the new machine. The report was circulated among many other researchers who incorporated key design features into their own projects. Examples of seminal features of the EDVAC design were serial operation of the machine, separate mechanisms for calculation and storage, a hierarchical arrangement of memories with a small amount of expensive fast storage and progressively more cheap slow storage and the use of a stored-program for instructions. Often the innovations of the Draft Report are called in summary “the stored-program concept,” after a single feature. The stored-program has been summarized as the storage of instructions and numeric data in the same format in the memory of an automatic calculator. [3] This feature makes it possible to perform a conditional branch by manipulating the instructions based on intermediate results. This in turn makes a machine Turing complete, that is with the addition of unlimited storage the machine would be a machine capable of carrying out any possible computation.

There are two things to note about this reduction. First that the stored- program is not the only way to achieve a Turing complete machine, such a machine could be achieved by a mechanism that stored data and instructions in completely different formats. Most obviously a machine with hardware implemented conditional branch and unlimited potential for rewritable storage of separate instructions and data, such as magnetic tapes could be Turing complete. The most striking example of the potential to achieve a universal machine by other means was Raùl Rojas' demonstration that German computer pioneer Konrad Zuse's Z3 machine could in principle perform any finite computation a Turing machine could, if it were given an infinite tape and indefinite intermediate storage for data. This is despite the Z3's incredibly limited repertoire of capabilities. The Z3 could perform basic arithmetic as directed by a long tape of instructions. It was not however a stored-program, rather the program was fixed with a small rewritable storage for numerical data. Also the Z3 could not perform a conditional branch, except to stop under certain conditions such as a divide by zero error. Rojas demonstrated that a branch can be simulated on such a machine by carrying out both paths of the branch and negating, via multiplication by zero, any contributions from the path that would not be taken by a genuine branching machine. [4]

The second thing to note is that as narrowly defined a stored-program machine need not have any resemblance to the machine described in the "Draft Report." A machine with the basic stored-program might have no hierarchy of memory possessing only one form of storage, such as say a rewritable tape, or might store and calculate with the same mechanism in a massively parallel system. Despite this, "von Neumann architecture" and "stored-program computer" are used interchangeably, for example Wikipedia has a single entry for both terms. [5] The equation of these two terms strips them of some important connotations. For example, the problem of delivering new instructions and data to the processing unit of a computer from its memory has become known as the "von Neumann bottleneck", a clear reference to the features of the design from the "Draft Report" rather than a feature of a narrowly defined stored-program machine. [3]

The difficulty of defining the essential characteristics of the computer is already widely noted in the literature. The 2000 anthology of history papers, The First Computers, embraces the ambiguity of the concept and contents itself with discussing a broad range of machines in the 1930s, 40s and early 1950s as the first computers, but not accepting any one of these as the singular first computer. [6] My aim in this essay is to expand on this point detailing some non-material aspects that separate computers from other calculating machines. While certain hardware characteristics may be necessary to create a machine with the capabilities of the modern computer, they alone do not constitute the defining essence of the machine. Rather the practices, applications and expectations that surround a machine play a key role in defining what it is.

2 The IBM SSEC a Stored-Program Computer?

Part of my inspiration for this paper has been my research on astronomer Wallace J. Eckert (no relation to Presper Eckert) and his work at Columbia University and IBM. At IBM, Eckert was associated with the development of only a few machines. One of these was the Selective Sequence Electronic Calculator (SSEC) dedicated in January of 1948. This machine sits right at the cusp of the era of the modern computer, but receives limited attention from histories. The SSEC stored instructions and data in the same format, could automatically manipulate instructions and used this feature to vary operation in response to intermediate results. Therefore the SSEC embodied the narrow definition of a stored program computer. Yet this feature often went unmentioned even in accounts that emphasized the narrow definition of the stored-program computer as a, or the, key element of the modern computer. Later machines would be cited as the first stored program computer. [7]

In light of the emphasis placed on the stored-program concept by the literature, I found it incongruous that this feature of the SSEC would be so often ignored. This led me to pay attention to those accounts that recognized the novel features of the SSEC, but gave reasons to deny it the status of a modern computer. The accounts of the SSEC by IBM historians such as Charles Bashe and Emerson Pugh suggested some reasons. The SSEC's use of relays for the bulk of its "fast" memory meant that its operating speeds were somewhat limited compared to later machines and even the earlier ENIAC, despite its use of electronic arithmetic to perform computations. Perhaps more importantly the SSEC had limited impact on later developments. IBM's first commercial computer the IBM 701 was based on von Neumann's computer at the Institute for Advanced Study. The only machine design to take direct inspiration from the SSEC machine was the IBM 650. [8]

Computer scientist and historian Brian Randell offered explicit justification for his placement of the SSEC. While noting that the SSEC was probably the first machine capable of arithmetic manipulation of stored instructions, what I call the narrow stored program concept, he denied it status as the first stored program computer. He cited its small electronic storage size and claimed it operated more in the style of a fixed program tape machine. In disqualifying the SSEC Randell in part makes reference to the need for electronic speed, since he allows that the Manchester SSEM test machine was the first computer, although not a practical one, despite the fact that its memory was only larger than the SSEC's with reference to the electronic storage and smaller when relay storage was included. [9]

However, it is clear that for Randell the stored program means more than the narrow concept and electronic speed. In introducing the stored program concept he noted its significance as allowing universal computation but added that its lasting significance consisted in the concept of using the computer to help prepare its own programs. Randell saw in the stored-program concept the seed of things like compilers and operating systems. Fixed program tape machines do not realize this sort of flexibility and the SSEC would have had trouble holding its full program in its memory. [9] Elsewhere Randell rejected the notion that merely storing instructions and data in the same part of the machine is sufficient to define the stored program concept, referring to the stored program as a practical engineering realization of Turing's universal calculator. [10] Randell's analysis argues that the stored-program

was not merely hardware, but a use and attitude towards the potential of that hardware.

Randell was not alone in viewing the stored program concept as richer than the bare storage of instructions and data together. The late Australian historian of computers, Allan G. Bromley, broke the origin of the "stored program concept" into 10 distinct stages of development. For Bromley the development began with the separate efforts to create digital electronic arithmetic and storage circuits and ended with the development of the idea of hierarchical ideas of machine programming, such as microprogramming with so-called firmware. Bromley saw the development of microprogramming by Maurice Wilkes in 1952 as the end of the development of the stored program concept. The storage of data and instructions in the same format occurs in the middles stages of Bromley's story, distinct from other developments, such as using separate parts of the machine for storage and processing and the development of compilers. [11] Like Randell, Bromley saw the stored program concept as more than mere hardware and more as an attitude towards machine use and design.

Now we come to the other inspiration of my paper: a discussion of the SSEC by French historian Georges Ifrah. In his Universal History of Computing, Ifrah recognizes the SSEC's stored-program nature, however he declares it to be the first "near-computer" in history. Ifrah gives two reasons to deny it full computer status. First its use of mixed electronic and relay technology that he argues was an anachronism and more importantly that the SSEC embodies "inconsistencies of logic." Ifrah argues that the designers of the SSEC lacked a clear theoretical understanding and vision for its operation and so made it unnecessarily complex. Ifrah gives von Neumann and the draft report credit for synthesizing the elements of the computer into a complete theory, while maintaining that in the history of technology there is neither a singular first inventor nor invention but a continuous series. [12]

Ifrah's suggestion that the SSEC design was unnecessarily complex has at least some justification. The machine used 12 500 expensive and unreliable vacuum tubes, whereas later machines would obtain more functionality with far fewer tubes. [7][8] Operators of the SSEC have had mixed opinions. John Backus, father of FORTRAN and an early SSEC programmer would later state that "I think it's an extreme stretch to consider it the first `stored program' computer." [13] However A. Wayne Brooke, the IBM engineer responsible for maintaining the SSEC, felt that quite the contrary the SSEC did everything a stored-program machine did just in different ways. [14]

Brooke spent a great deal of time in the early 1980s drafting a description of the SSEC and arguing for its status as first stored-program computer. The drafts of this unpublished account reveal many interesting aspects to the machine, including the creation of a special modification to facilitate conditional branch. The SSEC did not have a specific operation for branching and worked instead by arithmetic manipulation of the instructions in the memory. The modification Brooke described allowed the branch to a subroutine to be carried out in a more automatic and straightforward manner. [15] This suggests that the concepts and means of control of the SSEC were in flux after its completion.

Another aspect of the SSEC that suggests its "near computer" status is that it lacked single fixed physical configuration and total control of capabilities by the program. Certain aspects such as access to relays, tape drives and the like were

determined by plugging. [8] While relatively minor compared to the extensive rewiring required in the original ENIAC, it suggests the lack of commitment to program control.

3 Essence of the Stored Program Concept

While I remain somewhat skeptical of Ifrah's broad strokes analysis I found its boldness inspiring. In honour of Ifrah's suggestion I take the title of this paper from a maxim of French philosopher, Jean-Paul Sartre. "Existence precedes essence." In this Sartre wished to indicate that the meaning of human life is not given but created by individual human deliberation and action. [16] In a parallel I want to agree with Ifrah that the mere existence of the necessary hardware does not confer the essence of the computer, rather the operators' use of the machine determines its status.

I propose that while the traditional elements of the von Neumann architecture (a large fast electronic memory separate from a processor, serial processing of instructions, a stored program and so on) were necessary for the modern computer they were not sufficient. What was required beyond these hardware elements was a regulating ideal that the machine should be as general purpose as possible and a realization that the instructions themselves were subject to manipulation and generation by the machine. If endless plugging of wires on the ENIAC had simply been replaced by endless hand coding of data and instruction numbers in later machines, then the improvement in power and flexibility of those machines would have been limited. Finally, a stored-program machine should have practical achievements that exemplify these characteristics and aims, not mere theoretical hardware capabilities.

It is illustrative to note that the concept of subroutine libraries were quickly implemented in the first stored-program computers developed. The concept was first proposed by von Neumann and Goldstine working on the EDVAC project. The EDSAC, completed and running in 1949 at Cambridge, was the first machine to employ such a library. The developers hoped to avoid error by using computer code known to be reliable and save time in writing code. They disseminated this technique by publishing a programming manual in 1951 that detailed many of these subroutines. [7] The Manchester Mark I, which became operational a little later in 1949, also made use of a subroutine library. This library was an important resource for those who bought the Ferranti Mark I computer based on the Manchester machine such as the University of Toronto. [17] Note that these subroutine libraries, while only directly applicable to identical machines, served as a means of distributing standard programming techniques. Rather than each new computer having to be operated in a completely new way, techniques could be accumulated.

The reuse of code exemplifies both the standardization that is a hallmark of the modern computer and the use of the machine to make programming easier. In this case the operators take advantage of the fact that computer instructions were encoded in machine readable and therefore machine reproducible form. This feature was hardly exclusive to the modern computer, but is still an important element of its

success. The importance of digital copying is even more prominent today as more and more media is distributed digitally in various ways.

Actual preparation of code by machine methods was slower in coming. One of the first programs written at Cambridge was called “Initial Orders.” This program converted programs written in programmer’s notation of the machine code into the actual binary code of the machine. [7] While a small step, such a scheme represented an understanding that the computer could aid in construction of its own programs. However, operators would have to wait until 1952 before development of more complex programming aids like programming languages began development. [3] Of course it was only in 1952 that computers began to become more widely produced. Therefore developments of these elements of the stored-program concept can be seen to occur relatively quickly. Programming languages also allowed programs to be distributed relatively directly across platforms and so served to unify machines that were still diverse in terms of hardware.

The importance of these elements can again be seen by comparison with the limited development of the SSEC. The SSEC made use of subroutines in its programs often storing them as actual loops of paper tapes in an extensive number of tape drives. However, engineer A. W. Brooke could recall no instance where subroutines were re-used in different programs in the four years the SSEC was in use. [15] The SSEC was never used to generate or check coding because of scarce machine time, but the IBM 604, a punched card calculator, was used to perform simple arithmetic checks and manipulations of the instructions. [18] Finally, just as the hardware of the SSEC left few descendants, the programming techniques also apparently had limited impact. The lack of standardized, or widely communicated, programming techniques in the IBM SSEC show how it failed to achieve the full potential of the stored-program. At the same time the lack of such success helps explain its rapid fall into relative obscurity.

4 The Protean Computer?

Historian Michael Mahoney made a similar claim about the nature of the computer when he suggested that the computer was a “protean” machine, its nature determined by the tasks set and programs written for it. According to Mahoney it has no nature to guide the way it is adopted as compared to other technologies. [19] Whether his position is different from mine or not, I would make a difference in emphasis on certain points. First Mahoney seemed to seat the protean nature of the machine in its mere hardware, but, as I explained above, hardware is not sufficient to achieve the requisite degree of generality. The computer finds wide application because people are ready to use it in that way. Perhaps more importantly it seems computers become ever more universal because that is one of the ambitions of their developers.

Components that would make a computer’s features too specific were eschewed by developers, ignored by users or viewed as troublesome, as in the problems associated with optimal coding of magnetic drum memories. In drum memories information was only available at one point in rotation, therefore the computer might have to wait for an entire revolution to retrieve a piece of data or instruction. Users often sought to

optimize these retrievals by distributing instructions and data so that minimal time was lost to waiting for the drum to rotate. [3] Such optimization was completely hardware specific though. Later memory technologies became more truly random access and generic.

Also, I wholeheartedly agree with Mahoney's emphasis on seeing how various users and communities adapt the computer to their use. However, this does not mean that the computer is inert. In fact, if the computer were truly inert then adoption would be a straight forward matter. The problem already mentioned of the "von Neumann bottleneck" illustrates one example of the difficulty in reshaping the stored-program computer. Adopting new calculating technology often leads to change in practices and goals. The fading away of mathematical tables as a resource in scientific computation provides a good example.

The SSEC was provided with extensive fast table look-up facilities, including dozens of potential tape drives for tables in paper tape form. This is understandable as Wallace Eckert who oversaw its design had a long history of using tables in his work in astronomy. He had pioneered the use of punched card machines in scientific computation at Columbia in the 1930s and made use of mathematical tables in punched card form. Eckert had chosen as the first problem for the SSEC a computation of some positions of the Moon. The position was given by the summation of a long series of trigonometric functions. In the original incarnation of the program high precision values of the sine and cosine function had been obtained from a table punched onto tape. However, the program for this calculation was later reworked and the high precision values of sine and cosine found via calculation of an arithmetic series. [20] With the rise of the electronic computer extensive mathematical tables would fade from use in complex scientific computation, computers could use such tables, but special provision was rarely made for their use as in the SSEC.

Mathematical tables had been a vital component of computation in subjects like mathematics where extended calculation was done. The investment in these methods existed not only in physical copies of the tables, but in the extensive methods of tabular interpolation developed by practitioners. [21] As mentioned at the beginning of this paper Leo Vernon had predicted in 1939 that the "dream machine" would be able to read in tables with lightning speed. Yet the use of tables or not was dictated by the economies of speed in calculation versus reading off values from storage. Electromechanical punched card machines could read faster than they could carry out arithmetic, but not so early electronic machines.

Also, while the computer's nature may be protean, individual computers were not. Mahoney acknowledged this when he gives a definition of the history of computing as "the history of what people wanted computers to do and how people designed computers to do it." [19] However, it bears emphasis that individual computers were designed to be better suited to some tasks over others and so were not protean. For example, in the 1950s IBM marketed some of its computers as scientific machines and others as business machines. Thus the first IBM computer, the IBM 701, was designated for science, while the IBM 702 was designated a business machine. As a result the IBM 701 used a pure binary representation to store and manipulate values, while the IBM 702 used binary coded decimal and this trend continued for several later iterations of machines at IBM. [8]

The distinction between a binary and decimal machine may seem of little import, since it is trivial to convert input and output. However, many would-be operators who had to program in machine code did not see it that way. Decimal was seen as more congenial to the less technically learned business users. Coding in binary seems to have been a worry even for some scientists. For example astronomer Paul Herget did his own programming on the decimal Naval Ordnance Research Calculator (NORC), but was unwilling to do the same on the binary IBM 704. He suggested the engineers have their pay made out in binary until they saw the error of their ways. [22] In principle a binary and decimal machine might not be much different because after all the stored-program meant they could perform all the same calculations. Again human practice helps define the nature of a machine beyond its hardware.

5 Conclusion

I have defined the modern "stored-program" computer as a machine that not only possess the hardware characteristics suggested by the "Draft Report on the EDVAC," but also was used in a certain way. Specifically the philosophy of use must take full advantage of the ability of the machine to generate its own instructions and strive for maximum generality. Finally the machine must also practically achieve significant generality. Ultimately my definition of the modern computer is qualitative rather than a strict demarcation. Therefore any claim to which machine is the first on this basis is problematic. Still, using it I have attempted to justify the common distinction made in the history of computing literature between the modern computer and their various precursors. The justification is that this definition better reflects the trends in the history of computing. In principle it might make no difference how we define things, but in practice our definitions help determine how we summarize and organize history.

Acknowledgments. Thank you to my mother and my colleague Isaac Record for proof reading versions of this paper. I also wish to thank the Special Collections of the library of the University of North Carolina at Raleigh, where I carried out research that has been integral to this paper. Finally I must thank my anonymous reviewers for their comments and corrections.

References

1. Vernon, L.: Tools for Brains. Astounding Science-Fiction July 1939, 80-90 (1939)
2 . Goldstine, H. H.: The Computer from Pascal to von Neumann. Princeton University Press, Princeton (1993)
3. Ceruzzi, P.: A History of Modern Computing. MIT Press, Cambridge, Mass. (1998)
4. Rojas, R. The Architecture of Konrad Zuse's Early Computing Machines. In: The First Computers: History and Architectures, pp. 237-262. MIT Press, Cambridge, Mass. (2000)
5. Wikipedia: von Neumann architecture, http://en.wikipedia.org/wiki/Stored_program_computer Accessed Febuary 9, 2010.

6. Rojas, R. and Hashagen, U.: The First Computers: History and Architectures. MIT Press, Cambridge, Mass. (2000)
7. Campbell-Kelly, M. and Aspray, W.: Computer a History of the Information Machine 2nd. Westview Press, Boulder, Colrado (2004)
8. Bashe, C. J., Johnson, L. R., Palmer, J. H. and Pugh, E. W.: IBM's Early Computers. MIT Press: Cambridge, Mass. (1986)
9. Randell, B.: The Origins of Digital Computers: Selected Papers, 3rd edition. Springer-Verlag, New York (1982)
10. Randell, B.: The Origins of Computer Programming. IEEE Annals of the History of Computing, 16 (4), 6-14 (1994)
11. Bromley, A. G.: Stored Program Concept: The Origin of the Stored Program Concept. Technical report 274, November 1985. Basser Department of Computer Science, University of Sydney: Sydney (1985). http://www.it.usyd.edu.au/research/tr/tr274.pdf Accessed April 28, 2010.
12. Ifrah, G.: The Universal History of Computing. John Wiley & Sons, Inc., New York (2001)
13. Backus, J. quoted in: The IBM Selective Sequence Electronic Calculator, http://www.columbia.edu/acis/history/ssec.html Accessed Febuary 12, 2010.
14. Brooke, A. W.: Letter to Lyle R. Johnson, Dec. 27, 1980. A. Wayne Brooke Collection, University North Carolina at Raleigh, special collections, MC#268, Box 1, Folder #2: Section 1.2 (1980)
15. Brooke, A. W.: SSEC, the first electronic computer. Manuscript. A. Wayne Brooke Collection, University North Carolina at Raleigh, special collections, MC#268, Box 1, Folder #9: Series 2.3 writings (undated)
16. Stanford Encyclopedia of Philosophy: Existentialism: Existence Proceeds Essence. http://plato.stanford.edu/entries/existentialism/#ExiPreEss Accessed Febuary 13, 2010.
17. Campbell, S.: The Premise of Computer Science: Establishing Modern Computing at the University of Toronto (1945-1964). PhD Thesis, University of Toronto, IHPST (2006)
18. Brooke, A. W.: The Hallowed "Stored-Program Concept". Manuscript. A. Wayne Brooke Collection, University North Carolina at Raleigh, special collections, MC#268, Box 1, Folder #11: Series 2.4 writings (undated)
19. Mahoney, M. S.: The histories of computing(s). Interdisciplinary Science Reviews, 30(2), 119-135 (2005)
20. Eckert, W. J., Jones, R. B. and Clark, H. K.: Construction of the Lunar Ephemeris. In Improved Lunar Ephemeris: 1952-1959, pp. 283-363. Washington, D. C., U.S. Government Printing Office, (1954)
21. Grosch, H. R. J.: Computer: Bit Slices from a Life. 1st ed. Third Millennium Books/Underwood-Miller, Novato, California (1991)
22. Dubcombe, R. L.: Early Applications of Computer Technology to Dynamical Astronomy. Celestial Mechanics 45, 1-9 (1989)