Jonathan von Neumann and EDVAC

Philip Levis

November 8, 2004

Abstract

We review the historical context in which John von Neumann published "The First Draft Report on the EDVAC." We examine previous systems, such as ENIAC, concurrent systems such as EDSAC, and examine how the ideas in the EDVAC report influenced computer design. We compare the computer described by the report with the one actually built, and note how abstractions differ from those in current computers. Examining the principles behind von Neumann's design, we note the intellectual difficulties and mistaken conceptions that are better understood today. Finally, we consider von Neumann's arguments for the utility of computing, and how he foresaw Moore's Law.

1 A Bit of History

John von Neumann was a genius. Born in Budapest on the December 28th, 1903, he earned a Ph.D. in Mathematics and a Ph.D. in Chemistry (from different institutions) at the age of 23 and joined Princeton's Institute of Advanced Study (where people such as Albert Einstein worked) at the age of 30. He had a tendency to study a field, publish a landmark paper, and move on; Roy Weintraub, a professor of economics at the University of Pennsylvania, called one of von Neumann's' two papers on economics, "the greatest paper in mathematical economics ever written." Von Neumann often had insights into the repercussions of work that others would understand later; on hearing Gödel present his results on formal incompleteness, he immediately forsook logic and said "it's all over."

In 1942, von Neumann joined the Manhattan Project's efforts to build a nuclear weapon. His mathematical abilities led him to the task of working on some of the most complex equations and calculations needed. The sheer degree of computation needed was immense: Oak Ridge had rooms full of people whose job was to operate mechanical calculators.

Other areas of the military had a great need for computation as well. In particular, the Ordinance Ballistic Research Laboratories (OBRL)of the U.S. Army needed to compute trajectory tables for artillery. The standard approach was to solve complex differential equations of motion – based on many variables, such as humidity, air speed, and inclination – with the Bush differential analyzer, a mechanical computing device. Generating a single table took four hours, and the OBRL needed lots of them.

The large amounts of computation needed led to the proposal and development of ENIAC, or Electronic Numeric Integrator and Calculator. Dr. John W. Mauchly and J. P. Eckert, Jr of the University of Pennsylvania proposed ENIAC to the BRL in 1943, and received funding to build it. ENIAC became operational in June of 1944, and its tenders slowly upgraded it with additional resources (such as a square root calculator).

ENIAC was designed and built quickly, as the war effort needed it greatly. However, during its construction, Mauchly and Eckert saw ways in which it could be greatly improved, and began designing its successor, the EDVAC (Electronic Discrete Variable Automatic Calculator). Both Mauchly and Eckert, but Mauchly in particular, saw great financial possibilities in electronic computers, believing that they would be in high demand after the war.

Although the armed forces were building computers to speed calculations, the people involved in the Manhattan project – including von Neumann with all of his security clearances – were unaware of the efforts. One day in August 1944, von Neumann was waiting on the Aberdeen train platform. Lieutenant Herman H. Goldstine, officer at the BRL (located in Aberdeen) involved in ENIAC, happened to be waiting on the same platform. Goldstine, who had heard von Neumann speak once, cautiously approached him. In their conversation, Goldstine mentioned that he worked on electronic computers; von Neumann was immediately interested, and began to interrogate him on the subject. Understanding the implications an electronic computer could have on sciences that were greatly limited by computation, von Neumann arranged to see the ENIAC, which was nearing completion.

After his brief discussions with the ENIAC/EDVAC group – he was far too busy to be a regular participant – von Neumann wrote ”First Draft of a Report on the EDVAC,” which sketched the workings of how he thought EDVAC should work. Correspondingly, he put only his name on it. Mauchly and Eckert, concerned with patents for post-war money-making, had so far kept computer design relatively secret. von Neumann sent his document to a few people, who began to disseminate it widely, such that von Neumann’s name was predominantly associated with EDVAC. As we’ll later see, this document led to the fragmentation and dissolution of the EDVAC group; the working EDVAC was built by a completely different set of people than its initial designers, and correspondingly differed greatly from what von Neumann described.

The principal leap forward in von Neumann’s report was the argument for a ”stored program” computer, in which the computer stored the operations to perform in its memory. Programming ENIAC required flipping switches and rewiring some of its parts: although this was relatively efficient, it could often take a quarter hour to reprogram ENIAC for a computation that only took thirty seconds. In contrast, EDVAC read in some kind of input that encoded its operations, stored them in memory, and executed from there. Eckert had proposed the idea almost a year earlier, but von Neumann seized upon it and argued for it fervently.

2 Computers of the Time

At first read, many of the hardware concepts in the first draft seem quite strange to someone familiar with modern computers. The relative resource costs of vacuum tubes

(versus modern transistors) dictated a very different kind of design. A single SRAM bit in modern computers, for example, takes six transistors. This would have been six vacuum tubes in 1945, so storing 128 bytes of memory would take 6,000 tubes, an exorbitant amount that would not even be considered.

The problem of storing temporary state was not a simple one. After several proposals, Eckert and Mauchley settled on mercury delay lines. Mercury delay lines take advantage of the electro-mechanical properties of quartz: applying a voltage to quartz causes it to oscillate, and oscillating a piece of quartz will cause it to produce a voltage. A delay line has a quartz crystal at each end of thin column of mercury. [1] Applying a voltage to the quart at one end of the column produces a wave; when the wave reaches the other end, it oscillates the quartz there, producing a voltage. This voltage can be amplified (to account for resistance and other factors) and fed back into the first crystal, storing the data as long as needed. At any point, a delay line has a long string of bits propagating through it.

Delay lines are not random access memory: if a computer needs a particular piece of data, it must wait for those bits to propagate to the end crystal. This requires precise timing, to know when a given series of bits will arrive. The speed that waves propagate, however, depends on the temperature of the medium. Correspondingly, EDVAC's delay lines had to be kept at a very stable temperature, approximately 40° C. This required huge temperature stabilization/regulation assemblies on the delay lines, one of the reasons that they constituted most of the space of the final machine. The cyclical nature of delay line storage means there's a tradeoff between the number of separate lines, the number of bits each stores, and the expected latency to be able to access a given value. For example, the final EDVAC design included many long delay lines for standard memory, as well as three very short lines for storing instruction operands.

By the early 1950's, magnetic drums were replacing mercury delay lines as the storage medium of choice. Like delay lines, magnetic drums have a non-random access pattern. However, they are much less expensive to produce and maintain.

3 The Report vs. EDVAC

"The First Draft Report on the EDVAC" presents several concepts present in modern day computers, but uses different terminology:

EDVAC	Modern Term
minor cycle	word
major cycle	memory bank
order	instruction
CA	ALU
M	MMU

The machine von Neumann proposed is a pure binary architecture: numbers are simple signed binary (the first bit denotes positive or negative). Based on what he expected as the reasonably expected degrees of precision, he concluded that numbers should be thirty-two bits. One section of the draft discusses in depth how to implement

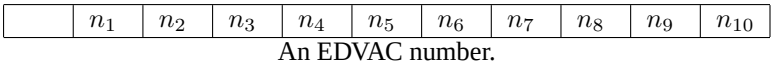
[1] Other substances besides mercury could be used, such as water, but mercury's properties made it best suited for the task

a pure binary adder; although the core circuit (an XOR, with an AND for the carry bit) is what one might expect from a modern architecture textbook, it introduces the notion of a delay along some of its lines.

The need for delays along some of the lines (e.g., the carry bit) is due to assumptions of the memory model. Unlike a modern ALU, which performs all of the binary operations in parallel on full words, von Neumann assumed mercury delay lines, which would deliver the bits of a minor cycle one at a time: the adder then produces the result one bit at a time, and needs to propagate the carry forward in time.

Unfortunately, the political repercussions of von Neumann's memo prevented many of its ideas from being fully realized. The EDVAC group fractured and dispersed shortly when the first draft emerged, and a new team of engineers were given the task of building the machine. The actual EDVAC built bears little resemblance to von Neumann's proposal. Additionally, completely replacing the engineering group greatly delayed EDVAC's development. Other computers built by different groups, such as the EDSAC and Manchester Machine in the U.K., emerged as working stored program computers before the EDVAC.

First, the EDVAC built was not a pure binary architecture. Instead, it stored numbers in BCD (binary coded decimal), where each decimal digit of a number is stored as four bits. Numbers are ten digits of precision, so forty bits long. Instructions have a uniform format: there are four opcode bits (twelve of the sixteen possible orders were used), and four ten bit operands. The first two operands are the memory addresses of the inputs to the order, the third operand is the memory address to put the output, and the fourth operand is the address of the next order to execute:



An EDVAC number.

An EDVAC order.

Although numbers were 10-digit BCD representations, addresses were pure binary: the ten bit operands addressed into the 2^{10} or 1024 minor cycles of EDVAC's memory, one eighth of the 8192 that von Neumann proposed.

4 Binary Arithmetic

The use of mercury delay lines, and their inherently temporal behavior, leads to some of the circuits proposed in the EDVAC report being rather convoluted. Additionally, although von Neumann saw the utility and efficacy of a purely binary architecture, his proposal does not acknowledge all of the ramifications of this approach.

For example, in his proposed order set, there are operations such as $\sqrt{}$, but no binary orders, such as shift and or. In the modern, post-RISC world, the idea of high-level instructions such as $\sqrt{}$ seems problematic; however, an architecture that can store 2^{10} words and instructions combined does not have the resource flexibility to implement square roots in software. The modern approach is to decompose an architecture into its

simplest parts, and then build complexity up. The neuron model allows gates such as OR and AND, but others such as XOR are absent.

Interestingly enough, the EDVAC report makes almost no mention of optimization. Beyond choosing vacuum tubes over relays, and selection of delay line dimensions, the performance effects of the rest of the design are left unsaid. This can be in part due to the incomplete and terse nature of the report; rather than defend his design, von Neumann was merely trying to state it. Additionally, performance is usually only meaningful in a comparative sense. Finally, good system design dictates building it first, and optimizing later. From the designs of other computers – such as the actual EDVAC built – it's clear that pure binary arithmetic wasn't quite yet understood. Even if von Neumann didn't grasp all of its implications, he grasped more of them than others did.

5 Computer Economics

History has numerous examples of computer scientists seeing only limited uses for their inventions: early computers were a common recipient of this sentiment. In the late 1940s and early 50s, computers were for large organizations or the government. The statement, "I think there is a world market for maybe five computers," although apocryphally attributed to Thomas Watson, articulates the sentiment of the times. The advent of rapid calculating machines raised the question: what were they good for? These machines could perform the work of hundreds of people.

To von Neumann, the question of what huge computational resources would be useful for was a meaningless one. He considered computing power in economic terms: as a resource, it should be couched in terms of supply and demand. Clearly there was demand for the current computers; as production became less expensive, demand would increase.

Some of von Neumann's early work (well before World War II) had established the notion of an expanding economic model (EEM); EEMs set aside the notion of first order inputs, such as "labor." Instead, by acknowledging that workers consume resources in order to produce more, an EEM establishes that economies can grow at a particular interest rate independent of their size. That is, economies can expand indefinitely, at an exponential rate (e.g., 3%).

Couching computers in economic terms – supply and demand – immediately invokes von Neumann's expanding model. A simplistic view of the model would say that the computational power of computers could be used to design more powerful and efficient computers. The details of growth aside, looking at computers in this light raises the notion of exponential growth.

Moore's Law, one of the most powerful characteristics of computers, states that the number of transistors on a chip will double every eighteen months. The literal notion of the law is very narrow; it has since grown in meaning to indicate the general behavior of exponential growth. Chips are twice as fast; disks are twice as large; networks have double the throughput; memory size doubles.

One could say that von Neumann's genius and effect on CS went beyond the stored program computer: he also presaged the driving law of growth behind the importance

of the computer industry today.