The Computer as von Neumann Planned It

M. D. Godfrey*
D. F. Hendry**

Address for correspondence:

Michael D. Godfrey
Stanford University
Information Systems Lab., rm 233
David Packard Bldg.
350 Serra Mall
Stanford, CA, 94305
email: godfrey@isl.stanford.edu

Published in: *IEEE Annals of the History of Computing,* vol. 15, no. 1, 1993, pp. 11-21.

Correction history:

26 February 2005: A clarification has been provided in Section 4.2 in order to explain the "inhibitor" symbol used in the inverter. The reference above to the published copy of this paper was corrected(!). And, since after this paper was published the edited "First Draft Report on the EDVAC" was published, a reference to this publication is provided in [1].

15 June 2006: The date of publication of the Weik Report [9] was corrected from 1951 to 1955.

1 December 2010: While correcting a few typographical errors I took the opportunity to rearrange the Tables, Figures and text to be more readable and more like the layout of the original publication.

---

* Information Systems Laboratory, Electrical Engineering Department, Stanford University, Stanford, CA.

** T-H Engineering, Inc., Altadena, CA.

The Computer as von Neumann Planned It

M. D. Godfrey
D. F. Hendry

ABSTRACT

We describe the computer which was defined in von Neumann's unpublished paper *First Draft of a Report on the EDVAC,* Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945. Motivation for the architecture and design is discussed, and the machine is contrasted with the EDVAC which was actually constructed.

Keywords: architecture, computer, EDVAC, stack, tagged-memory, von Neumann architecture.

1. Introduction

John von Neumann made a key contribution to the understanding and development of computer architecture and design in his unpublished report titled *First Draft Report on the EDVAC* [1]. However, in reading work which refers to this report and to the EDVAC (the acronym is defined in [10] to be: **E**lectronic **D**iscrete **VA**riable **C**omputer) computer which it described some perplexing observations emerge:

1. The constructed EDVAC is usually described as being based on the von Neumann Report [1].
2. The von Neumann Report is often described as the collective work of the Moore School group, unfairly given the sole authorship of von Neumann (see, for example, page xv of [11]). This would suggest that many of the ideas in the Report were shared by the Moore School design group and therefore would be expected to appear in the constructed machine.
3. The EDVAC has been described on numerous occasions, but these references do not agree about basic facts: For example, a key feature of any computer is the size of each *word* in the addressable memory. On this subject, Goldstine [2] indicates 40 bits, others (Burks [3]) state 32 bits. The only known publication giving the

correct value (44 bits) is Knuth [6]. The BRL Report [9] (which is well-known but was never published) also has the correct value.

Some of the evident confusion stems from the failure to distinguish between the "EDVAC" as described by von Neumann in the Report, and the "EDVAC" as constructed at the Moore School. While copies of the von Neumann Report were informally circulated at the time it was written, the Moore School design documents were kept private and were in fact classified and marked "CONFIDENTIAL." ([10] was changed to "unclassified" in 1947.) The confusion has been aggravated by the fact that von Neumann's Report has been reprinted only in incomplete or inaccurate forms.

The main purpose of this paper is to present the architecture given in the von Neumann Report in a form which is accessible to a wider audience, and to translate into modern terminology the formal machine definition given in the Report. We also compare this definition to the definition of the constructed EDVAC system. In doing this we hope to clarify important, but previously unrecognized, features of the von Neumann design and to clarify a number of the confusions that have arisen over the years. The most substantial description of the Moore School EDVAC is given in [10]. The Sections of [10] which specify the Moore School EDVAC in detail were written by Harry Huskey, who has also been most helpful in several discussions about the work at the Moore School. The accompanying paper [13] in this issue reviews the actual performance of the single constructed EDVAC that was delivered to BRL.

However, this paper in no way replaces the original von Neumann Report. Our purpose is only to make clear the definition of the EDVAC machines, and to clarify the origins of these definitions. The von Neumann Report contains a wealth of insight and analysis which is still not available elsewhere. Few people have had the opportunity to read and decipher the original typescript. One person who has understood the von Neumann design, particularly from the programming standpoint, is Donald Knuth. His paper [6] describes the main features and instruction code of the von Neumann design, and also discusses improvements that von Neumann developed after he had drafted the Report. It was von Neumann's intent that these improvements (most prominently a 32-word register file to replace the 3-register stack) should be incorporated into the Report. This was never done. The improvements were based on results from a sort program which von Neumann wrote to test the effectiveness of his design. A prominent feature of the Report is von Neumann's recognition that his computer would not perform relatively efficiently on sorting problems. This remains, substantially, an unsolved problem to this day.

Unfortunately, reading the Report is made difficult due to the incomplete draft form of the original, and the propagation of accumulating errors in the versions that have appeared in print [4, 7, 11]. These reprints have carried over the original errors, and introduced new errors. Since one reprinting [11] was based on a previous reprint [7], rather than on the original, further compounding of typographical errors has occurred. The incomplete copy in [4] is not very useful as it only includes the first 5 introductory Chapters. The inaccurate copies in [7] and [11] make von Neumann's original intent quite hard to discover mainly because of numerous mistakes in the mathematical notation. A typical paragraph in [7] (second paragraph, pg. 239) reads:

> Thus each DLA organ has now a number $\mu = 0, 1, \ldots, 255$ (or 8-digit binary), and each minor cycle in it has a number $\mathrm{p} = 0, 1, \ldots, 31$ (or 5-digit binary). A minor cycle is completely defined within M by specifying both numbers i, p. Due to these relationships we propose to call a DLA organ a *major cycle*.

This should have read:

> Thus each DLA organ has now a number $\mu = 0, 1, \ldots, 255$ (or 8-digit binary), and each minor cycle in it has a number $\rho = 0, 1, \ldots, 31$ (or 5-digit binary). A minor cycle is completely defined within M by specifying both numbers μ, ρ. Due to these relationships we propose to call a DLA organ a *major cycle*.

A few pages later in [7] (page 242) the notation switches from μ to u, but then later (page 243) p is switched to ρ since the typist stopped typing p and wrote in ρ by hand. The fact that the first reprint [4] contains only the first 5 of 15 Chapters has led to additional confusion. For example, the March 1992 issue of *Computing Reviews* contains a review of William Aspray's *John von Neumann and the Origins of Modern Computing*. The reviewer states: "(Perhaps the lack of publication accounts for discrepancies between the author's quotes and the version of the Report appearing on pages 355-364 of a book edited by Brian Randell... " The reviewer is obviously unaware of the fact that Aspray was referring to the full 15-chapter Report, not the 5 chapters reprinted in Randell [4]. The reviewer goes on to draw further conclusions about von Neumann's role in computer development based on the belief that the Report contained only the 5 introductory chapters, when in fact all of the substance of the EDVAC design and architecture as expressed by von Neumann is contained in Chapters 6 through 15.

The original manuscript from the Moore School [1] is easier to read than the published versions since it has fewer errors and it is easier to identify obvious typographical mistakes. However, as a first draft, it contains a great many typographical errors, particularly in the mathematical and special symbols. I have prepared a corrected version which reconstructs what was surely von Neumann's intended text. This version has not been published, but it is hoped that this will be possible in the future. In doing this work I converted the manuscript to TEX form so that it could be easily managed and made ready for publication. This also had the effect that this version is easier to read because of the improved typography. (Note added 26 February 2005: This version has been published. See [1].) Needless to say, the Report is a brilliant piece of work. All contemporary computer projects made use of material from the Moore School, typically including a copy of the Report. Alan Turing [12] explicitly based his computer design on the Report. However, curiously, the computer built under von Neumann's guidance at the Institute for Advanced Study did not follow the architecture or design principles of the von Neumann EDVAC. This fact deserves further study. (It is of course well-known that von Neumann's focus of interest had by then moved to other subjects, including new work on computer theory.)

We hope that both the availability of a corrected text and this introductory guide will make this key contribution more accessible.

2. The Two EDVAC's

That the EDVAC described by von Neumann (to be referred to as vN-EDVAC) and the EDVAC constructed at the Moore School (to be referred to as M-EDVAC) are very different in architecture and design will become clear below. It would be interesting to know how these differences arose, especially since the IAS machine [8] was closer to M-EDVAC than it was to vN-EDVAC. Von Neumann did not, after an initial period, get along very well with some members of the Moore School group due to both technical and other disagreements. It would appear that he wrote the Report as an effort to state the architecture and design as he imagined it at that time. The Report was apparently written while von Neumann was at Los Alamos and delivered to the Moore School in handwritten form. (Goldstine is reported to have said that he had a copy of the handwritten draft, but no such copy has been found in his archives at Hampshire College.) It was typed at the Moore School, but there is no evidence that von Neumann proof-read the result. The Report in any case had little ultimate impact on Eckert and Mauchly and the rest of the Moore School design team who designed M-EDVAC as they wanted it. (This account is based on conversations with Harry Huskey.)

M-EDVAC was a serial, synchronous, 44-bit word, 4-address (three operand addresses and the next instruction address), binary machine with 12 operation codes. It had 4 registers, but these do not appear to have been addressable. It used parallel comparison of the two arithmetic units for error checking. (See Williams [13] for details about the realized reliability and performance of the machine.) These and other features of the machine are summarized in Table 1 which is an excerpt from a BRL report by Weik[9].

vN-EDVAC was a serial, synchronous, 32-bit word, zero-address, binary machine with a hierarchical operation-code structure of 8 basic codes, 10 sub-codes, and one modifier. It had 3 non-addressable registers, organized as a *stack* mechanism. Tagged memory was used to distinguish instructions from data. This feature is more fully explained in Section 3.4.

The main features of the two designs are compared in Table 2.

Table 1: M-EDVAC

EDVAC

Specifications from Martin Weik, BRL Report No. 971.

Manufacturer

Moore School of Electrical Engineering
University of Pennsylvania

Operating Agency

U.S. Army Ordnance Corps Ballistic Research Lab, APG

General System

Applications: solution of ballistic equations, bombing and firing tables, fire control, data reductions, related scientific problems.
Timing: Synchronous
Operation: Sequential
A general purpose computer which may be used for solving many varieties of mathematical problems.

Numerical System

Internal number system: Binary
Binary digits per word: 44
Binary digits per instruction: 4 bits/command, 10 bits each address
Instruction per word: 1
Total no. of instructions decoded: 16
Total no. of instructions used: 12
Arithmetic system: Fixed-point
Instruction type: Four-address code
Number range: $-(-2^{-43}) \leq x \leq (1-2^{-43})$

Arithmetic Unit

Add time (including storage access): 864 microsec (min 192 max 1536)
Multiply time (including storage access): 2880 microsec (min 2208 max 3552)
Divide time (including storage access): 2930 microsec (min 2256 max 3600)
Construction: Vacuum tube and Diode gates
Number of rapid access word registers: 4
Basic pulse repetition rate: 1.0 megacycle/sec
Arithmetic mode: Serial

Storage

Media	Words	Microsec Access
Mercury Acoustic Delay Line	1024	48-384
Magnetic Drum	4608	17,000

Includes relay hunting and closure. The information transfer to and from the drum is at one megacycle per second. The block length is optional from 1 to 384 words per transfer instruction.

Input

Media	Speed	
Photoelectric Tape Reader	942	sexadec chars/sec
	78	words/sec
Card Reader(IBM)	15	rows/sec
	100	cards/min

Output

Media	Speed	
Paper Tape Perf.	6	sexadecimal chars/sec
	30	words/min
Teletypewriter	6	sexadecimal chars/sec
	30	words/min
Card Punch(IBM)	100	cards/min
	800	words/min

Table 1-cont: M-EDVAC

Number of Circuit Elements

Tubes:	3563	
Tube Types:	19	
Crystal diodes:	8000	
Magnetic elements:	1325	(relays, coils and trans)
Capacitors:	5500	approx.
Resistors:	12000	approx.
Neons:	320	approx.

Checking features

Fixed comparison—Two arithmetic units perform computation simultaneously.
Discrepancies halt machine.
Paper tape reader error detection.

Physical Factors

Power Consumption: Computer,	50 kW
Space occupied: Computer,	490 sq. ft.
Total weight: Computer,	17,300 lbs.
Power consumption: Air Cond.,	25 kW
Space occupied: Air Cond.	6 sq. ft.
Total weight: Air Cond.	4345 lbs.
Capacity: Air Cond.	20 tons

Manufacturing Record

Number produced: 1
Number in current operation: 1

Cost

Rental rates for additional equipment:

IBM card reader	$82.50
IBM card punch	$77.00

Approximate cost of basic system: $467,000

Personnel Requirements

Daily Operation: 3 8-hour shifts. No. of Tech: 8.
7 days/week
No engineers are assigned to operation of the computer, but are used for design and development of improvements for the computer. The technicians consult with engineers when a total breakdown occurs.

Reliability and Operating Experience

Average error free running period: 8 hours
Operation ratio: 0.79. Good time: 130.5 hrs.
(Figures for '55) Attempted to run: 166 hrs./wk.
No. of different kinds of plug-in units: 3
No. of separate cabinets (excluding power and air cond.): 12
Operating ratio figures for 1954:
Operating ratio: 0.79. Good time: 129 hrs. Attempted to run: 163 hrs./wk.

Additional Features and Remarks

Oscilloscope and neon indicator for viewing contents of any storage location at any time.
Exceed capacity options: halt, ignore, transfer control, or go to selected location.
Unused instruction (command) halt.
Storage of previously executed instruction and which storage location it came from, for viewing during code checking.
Storage of current instruction and storage location from which it originated.
Address halt when prescribed address appears in any of 4 addresses of instruction to be executed by computer.
Tape reader error detection.

Table 2: vN-EDVAC vs. M-EDVAC

Feature:	vN-EDVAC	M-EDVAC	
	Basic Design		
Timing:	Synchronous	Synchronous	
Operation:	Sequential	Sequential	
	Numerical System		
Internal number system:	Binary	Binary	
Binary digits per word:	32	44	
Data bits per word:	31	32	
Memory tag bits:	1	0	
Bits/command:	3+5	4	
Binary digits per address:	13	10	
Instructions per word:	1	1	
No. of instructions decoded:	8+16	16	
No. of instructions used:	8+10	12	
Arithmetic system:	Fixed-point	Fixed-point	
Instruction type:	Zero-address code	Four-address code	
No. of registers:	3 (non-addressable)	4 (non-addressable (?))	
Number range:	$-(-2^{-30}) \leq x \leq (1-2^{-30})$	$-(-2^{-43}) \leq x \leq (1-2^{-43})$	
	Storage		
Media	Words	Words	Microsec Access
Mercury Acoustic Delay Line	8192	1024	48-384
Magnetic Drum		4608	17,000
	Number of Circuit Elements		
Tubes:	2000-3000 (est)	3563	
Tube Types:		19	
Crystal diodes:		8000	
Magnetic elements:		1325	(relays, coils and trans)
Capacitors:		5500	approx.
Resistors:		12000	approx.
Neons:		320	approx.

3. vN-EDVAC Architecture

Throughout the Report von Neumann mentions the need to develop the structure of the system giving consideration to both design and architecture issues. The interaction of time and space and the need for locality in time and space are repeatedly discussed. These issues arise particularly in the determination of the size and performance of the delay line memory and in choices of primitive operations.

One may subdivide architecture into standard categories: addressing, instruction definition, protection, interrupt control, and input-output. Only one aspect of the last three categories is defined in the Report. Instruction memory (words tagged as containing instructions) was protected against modification of any fields except the address field. It is not explained how memory could be initially loaded with instructions. However, this was presumably a part of the I/O system.

3.1 Addressing Structure

All address values are included in the load, store and control transfer instruction fields or are based on the value of the instruction address register (PC). Address modification is carried out by computation of the desired address and then storing the address into the address field of the appropriate instruction in memory. All addresses are given as a variable pair $[\mu, \rho]$ but this is purely for design reasons. All addresses are 13-bit, word addresses.

3.2 Number Representation and Arithmetic Operation

All numeric data (termed standard numbers) are 31-bit signed binary integers. The rightmost (first) bit in the 32-bit word is the tag bit, with zero meaning that the word contains a standard number. (Memory locations are taken to be increasing to the left.) Data are stored least significant bit first, sign bit following (to the left of) the most significant bit and with the binary point taken to be between the most significant and the sign bit. Negative numbers are in two's-complement form. Thus, the range of standard numbers is $-1 \leq n < 1$ with a precision of approximately 8 decimal digits. At this point, and throughout the Report, despite a couple of switches of notation, von Neumann is clearly a "little-endian" (See [5]). All data are arranged so that the least significant bit is "first."

Standard twos complement arithmetic is provided for addition, subtraction, multiplication, division, and square root. Rounding is provided by computing an additional check bit and "rounding to the nearest odd digit." This was done to avoid carry due to rounding. No provision is made for detecting out-of-range results for addition, subtraction or division.

3.3 The Central Arithmetic (CA) Unit

The CA contains three registers: I_{ca}, J_{ca}, and O_{ca}. I_{ca} is the input register and may be viewed as the top-of-stack register. J_{ca} is the second word of the stack, but may also be the source register for transfers within CA. O_{ca} receives the output of operations which use I_{ca} and J_{ca} as inputs. It always acts as an accumulator, i.e. all results are formed by:

result + $O_{ca} \rightarrow O_{ca}$.

However, the store operations optionally allow clearing of O_{ca} after the store operation. All store operations store O_{ca}. The interconnection of these registers is shown in Figure 1.

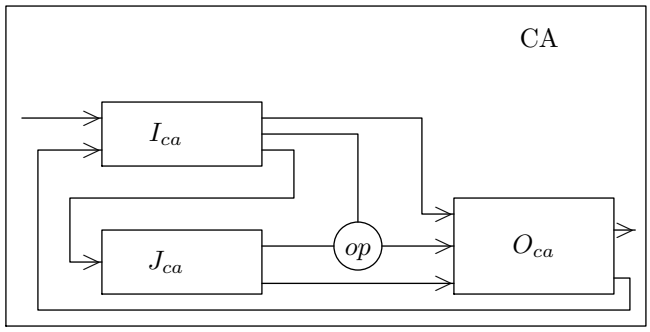


Figure 1: CA Diagram

This Figure is a more complete version of Figure 17 in the Report. The only way in which data enter the CA is by being loaded into I_{ca}. This always causes the previous contents of I_{ca} to be pushed into J_{ca}. The only path for data out of the CA is from O_{ca}. Operation of the binary operators, then, involves a sequence of the form:

LOAD	M[j]	$I_{ca} \rightarrow J_{ca}$, $M[j] \rightarrow I_{ca}$
LOAD	M[k]	$I_{ca} \rightarrow J_{ca}$, $M[k] \rightarrow I_{ca}$
OP		
STORE	M[r]	$O_{ca} \rightarrow M[r]$, optionally clear O_{ca}

In operations such as the above, I_{ca} and O_{ca} are implicitly addressed, as in stack-based or zero-address systems. However, instructions are also available to cause the transfers:

$I_{ca} \rightarrow O_{ca}$,
$J_{ca} \rightarrow O_{ca}$,
$O_{ca} \rightarrow I_{ca}$,

as indicated in Figure 1. Thus, for example, a program segment to compute

$$S = \sum_{i=1}^{4} x_i y_i$$

for literal data x_i and y_i could be:

0	Load zero.
$\times$	Clear accumulator.
x_1	Implicit load immediate x_1
y_1	Implicit load immediate y_1
$\times$	Multiply and accumulate.
x_2	
y_2	Repeat
$\times$	for
x_3	
y_3	remaining data.
$\times$	
x_4	
y_4	
$\times$	
$\rightarrow M[addr(S)]$	Store result at address of S.

A more fully parametrized procedure for computing inner products could be constructed using data address computations and loop control constructs. The lack of any address indexing mechanism or index registers causes array referencing to require additional instructions, as is true of many "modern" RISC (Reduced Instruction Set Computer) designs.

3.4 Instruction Definition

The definition and operation of the Central Control (CC) and Central Arithmetic (CA) sections of the vN-EDVAC are described in Chapters 11 and 13 through 15 of the Report. A full understanding of these Chapters requires some effort. Table 3 provides a glossary of the main components of the processor:

Table 3: Glossary

CA	Central Arithmetic-Logic Unit
CC	Central Control Unit
PC	Program Counter (address of current instruction)
SG	Switching and Gating Unit
A	$\boxed{\text{dl}}$ feedback amplifier
E-element	gate
M	Memory
R	External storage
I	Input channel
O	Output channel
L	Memory read (L_o) and write (L_i) lines
s	$\boxed{\text{dl}}$ select line
$\boxed{\text{dl(k)}}$	k unit delay
$\boxed{\text{M}_1}$	1-bit memory
$\boxed{\text{lk}}$	k-bit (serial) memory
minor cycle	32-bit word
major cycle	32 words of memory

The CC section is based on a conventional instruction sequencing mechanism for normal instruction processing. Given that the address in PC points to the current instruction in memory, the instruction at that address is fetched, decoded, executed, PC is incremented, and the operation cycle is repeated. (Note that M-EDVAC loaded the next PC value from a field in the current instruction word as is common in many microcode systems.) There are two exceptions to this standard instruction processing loop. First, is a control transfer instruction which loads PC with a new address. von Neumann discussed the possibility of an execute-remote operation as a "transient transfer," but decided against implementation. Second, if when an instruction word is fetched, it is found that the instruction tag bit is clear an implicit load-immediate instruction is executed. Thus, the contents of the word addressed by PC are loaded into I_{ca} and the PC is incremented in the normal way.

In the Report, the operations within CA and the load and store operations are described first. These (termed the *unpooled* orders) are **not** the actual machine instructions, but are distinct functional components of the instructions. Tables 4 and 5 summarize the notation and meaning for the unpooled orders. The actual instructions are termed the *pooled* orders and are summarized in Table 6. As can be seen, there are 8 instruction types (t'). The CA instructions use the sub-code field w, and the CA-store instructions use w and the modifier c. The main reason given for using the pooled orders as the actual machine instructions was the improved bit utilization in the instruction fields. As can be seen by comparing the unpooled orders in Table 5 with the pooled orders in Table 6, the only lost functions are δ, ϵ, and θ taken as separate operations. Since those operations would normally follow an α operation, the pooling seems natural. While the bit utilization of instruction words is still quite low (the minimum number of unused bits is 10), von Neumann remarks that code space is likely to be small as compared to data space, and room should be left for expansion of the address field. Such reasoning and foresight would have been helpful to recent and current microprocessor designs. Table 6 is arranged using the layout that might have been used by von Neumann, since he generally referred to the fields in the instructions using a layout of least significant bit at the right. (However, for numerical data, the 31 bits $i_1 ... i_{31}$, were usually referred to in left to right order even though they were stored "least significant bit first.") The instruction formats could equally have been drawn with the bit order reversed so that they would read more naturally from left to right. However, this arrangement emphasizes the bit-serial, "little-end" first structure of the machine.

Table 4: Notation for Instruction Fields

i_0	- tag bit
	0 - data
	1 - instruction
t	- unpooled instruction code
t'	- pooled instruction code
w	- operation sub-code for CA (stack) operations
c	- modifier for store operations
	0 - clear O_{ca}
	1 - retain value in O_{ca}
μ	- major cycle (delay line) address
ρ	- minor cycle (word within delay line) address

The two means of carrying out load-immediate operations deserve a comment. First, if a data word ($i_0 = 0$) is encountered during instruction processing an implied load of the contents of the word is carried out. In addition, the γ instruction (the second instruction in Table 6) loads the word which immediately follows it. It was not mentioned that γ should also have the side-effect of incrementing the PC an extra time so that the following word is not subsequently executed as an instruction. However, γ would appear to be almost entirely redundant since, if the following word is data ($i_0 = 0$) then just executing the following word as an implied load-immediate would have exactly the same effect as γ. It does not appear that von Neumann considered the possibility of following γ with an instruction word ($i_0 = 1$) so that instructions could be loaded by this means.

Unconditional control transfer is provided by the ζ instruction. Conditional transfers use the s operation (sign test) to select which address value will be moved to O_{ca}. This address must then be stored into the address field of an immediately following ζ instruction if an immediate transfer is to be made. However, the store could be made into a subsequent location in order to achieve some of the effects (and side-effects) of the delayed branch (RISC) operation.

Table 5: Operation code Definitions

1. Unpooled types (t):

order	name	definition
α	CC operations (stack)	See 2. below
β	load	$M[\mu, \rho] \rightarrow I_{ca}$
γ	load immediate	$M[PC + 1] \rightarrow I_{ca}$
δ	store	$O_{ca} \rightarrow M[\mu, \rho]$
ϵ	store immediate	$O_{ca} \rightarrow M[PC + 1]$
θ	CC move (stack)	$O_{ca} \rightarrow I_{ca}$
ζ	load PC (jump)	$M[\mu, \rho] \rightarrow PC$
η	I/O	not defined

2. α operations:

modifier	value	operation	definition
w =	0	$+$	$(I_{ca} + J_{ca}) + O_{ca} \rightarrow Oca$
	1	$-$	$(I_{ca} - J_{ca}) + O_{ca} \rightarrow Oca$
	2	$\times$	$(I_{ca} \times J_{ca}) + O_{ca} \rightarrow Oca$
	3	$/$	$(I_{ca}/J_{ca}) + O_{ca} \rightarrow Oca$
	4	$\sqrt{}$	$\sqrt{I_{ca}} + O_{ca} \rightarrow Oca$
	5	i	$I_{ca} \rightarrow Oca$
	6	j	$J_{ca} \rightarrow Oca$
	7	s	perform i or j depending on sign of Oca
	8	db	decimal $\rightarrow$ binary
	9	bd	binary $\rightarrow$ decimal

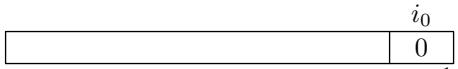
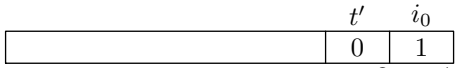
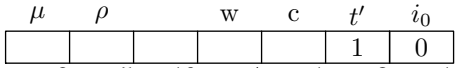
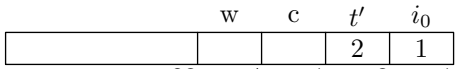
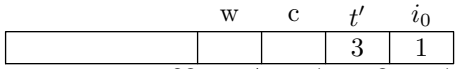
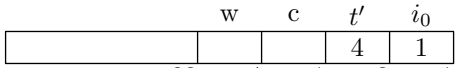
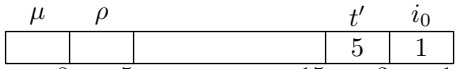
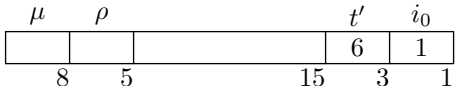
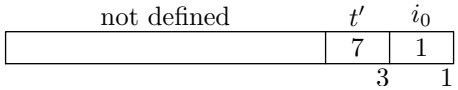
3. Memory addressing:

$\mu[\mu_7 ... \mu_0]$	major memory cycle (segment)
$\rho[\rho_4 ... \rho_0]$	minor memory cycle (word)
$[\mu, \rho]$	13-bit memory address (word-address)

Note: For store operations, if $i_0 = 1$ at the target address $M[\mu, \rho]$ then only the $[\mu, \rho]$ field is replaced by the high-order 13 bits of the operand.

Table 6 shows the organization of the pooled orders. Table 7 is taken directly from the von Neumann Report [4]. It shows the manner in which von Neumann summarized the logical definition of the machine.

Table 6: Pooled Orders

type (t')	instruction format (least significant bit at right, bit width of each field indicated below each instruction)	definition	meaning
$i_0 = 0$		$M[PC] \rightarrow I_{ca}$	load immediate
γ		$M[PC+1] \rightarrow I_{ca}$	load immediate
$\alpha + \delta$		OP; $O_{ca} \rightarrow M[\mu, \rho]$	stack operation; store
$\alpha + \epsilon$		OP; $O_{ca} \rightarrow M[PC+1]$	stack operation; store immediate
$\alpha + \theta$		OP; $O_{ca} \rightarrow I_{ca}$	stack operation; load
α		OP	stack operation
β		$M[\mu, \rho] \rightarrow I_{ca}$	load
ζ		$M[\mu, \rho] \rightarrow PC$	control transfer
η		I/O	input-output

Note: Von Neumann did not assign numeric codes to the 8 pooled orders. (At the end of the Report he indicated that he would do that next.) I have filled in numeric codes in the t' field in this Table just to make it more definite. Also, c is the O_{ca} clear flag as defined in Table 4, and w is the CA order modifier as defined in Table 5. The CA processing flow is shown in Figure 1 which is adapted from Figure 17 in the Report.

Table 7: Instruction Definition

<table>
<tr><th>(I)
Type</th><th>(II)
Meaning</th><th>(III)
Short Symbol</th><th>(IV)
Code Symbol</th></tr>
<tr><td></td><td></td><td></td><td>Minor cycle
$I = (i_v) =$
$(i_0 i_1 i_2 \cdots i_{31})$</td></tr>
<tr><td>Standard
Number
or Order
(γ)</td><td>Storage for the number defined by $\xi = i_{31} i_{30} \cdots i_1 = \sum_{v=1}^{31} i_v 2^{v-31}$ (mod 2) $-1 \leq \xi < 1$. i_{31} is the sign: 0 for +, 1 for −. If CC is connected to this minor cycle, then it operates as an order, causing the transfer of ξ into I_{ca}. This does not apply however if this minor cycle follows immediately upon an order w → A or wh → A.</td><td>Nξ</td><td>$i_0 = 0$</td></tr>
<tr><td>Order
$(\alpha)+(\delta)$
———
Order
$(\alpha)+(\epsilon)$
———
Order
$(\alpha)+(\theta)$
———
Order
(α)</td><td>Order to carry out the operation w in CA and to dispose of the result. w is from the list of 11.4. These are the operations of 11.4, with their current numbers w.decimal and w.binary, and their symbols w:
<table>
<tr><th>w.decimal</th><th>w.binary</th><th>w</th><th>w.decimal</th><th>w.binary</th><th>w</th></tr>
<tr><td>0</td><td>0000</td><td>+</td><td>5</td><td>0101</td><td>i</td></tr>
<tr><td>1</td><td>0001</td><td>−</td><td>6</td><td>0110</td><td>j</td></tr>
<tr><td>2</td><td>0010</td><td>×</td><td>7</td><td>0111</td><td>s</td></tr>
<tr><td>3</td><td>0011</td><td>÷</td><td>8</td><td>1000</td><td>db</td></tr>
<tr><td>4</td><td>0100</td><td>$\sqrt{\ }$</td><td>9</td><td>1001</td><td>bd</td></tr>
</table>
h means that the result is to be held in O_{ca}. → $\mu\rho$ means that the result is to be transferred into the minor cycle ρ in the major cycle μ; → f, that it is to be transferred into the minor cycle immediately following upon the order ϵ; → A, that it is to be transferred into I_{ca}; no →, that no disposal is wanted (apart from h).</td><td>w → $\mu\rho$
or
wh → $\mu\rho$
———
w → f
or
wh → f
———
w → A
or
wh → A
———
wh</td><td>$i_0 = 1$</td></tr>
<tr><td>Order
(β)</td><td>Order to transfer the number in the minor cycle ρ in the major cycle μ into I_{ca}.</td><td>A ← $\mu\rho$</td><td></td></tr>
<tr><td>Order
(ζ)</td><td>Order to connect CC with the minor cycle ρ in the major cycle μ.</td><td>C ← $\mu\rho$</td><td></td></tr>
</table>

4. vN-EDVAC Design

Von Neumann developed both the architecture and the design of the vN-EDVAC based on detailed analysis of the performance and resource requirements of a number of computational problems. Normal instruction sequencing was intended to permit instruction execution at the rate at which data arrived from the output of a delay line. The length of the delay line was determined based on the assumption that the average delay for a memory reference after an arithmetic operation would be short as compared to the arithmetic time. The ability to retain intermediate results in the CA registers reduces the frequency of store and load operations which would, unless addresses were carefully chosen, take an average of half a major (delay line) cycle time.

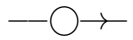
4.1 Memory Design

The intended memory was to be made up of mercury delay lines. Each delay line contained 32 32-bit words. At a clock rate of 1Mhz the circulation time of the delay line was about 1ms. The size of the delay lines was determined from the fact that 1ms was approximately the arithmetic time of the CA unit. Thus, new operands would become available from the current delay line at about the time they would be needed by the CA. The spirit of this analysis was sound, but it neglected important factors including instruction fetch requirements.

The size of the memory was determined after consideration of several possible numerical problems. In the course of the analysis of memory size and logic complexity, von Neumann remarks: "the decisive part of the device, determining more than any other part its feasibility, dimensions and cost, is the memory." Based on this analysis, von Neumann settled on a total memory size of 8k words or 256 delay lines.

4.2 E-elements and Logic

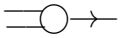
This is, as far as I am aware, the first substantial work (since Babbage) which clearly separated logic design from implementation, and gave a formal scheme for logic representation. It is a curious fact that the notation which was fully established and extensively used here was totally absent from subsequent works, particularly [8] and [10]. The simplest gate was drawn as



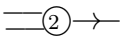
while an inverter was drawn as

Note that the —• symbol is an "absolute" inhibitor. See Section 6.4 of [1].

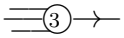
A two input OR gate would be



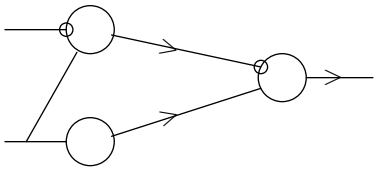
The notation



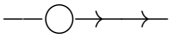
was used for a two input AND gate. A three input AND gate was given by



Thus, the number inside the circle indicated the minimum number of active inputs required to drive the output active. It is noted that a two input AND gate would be defined by

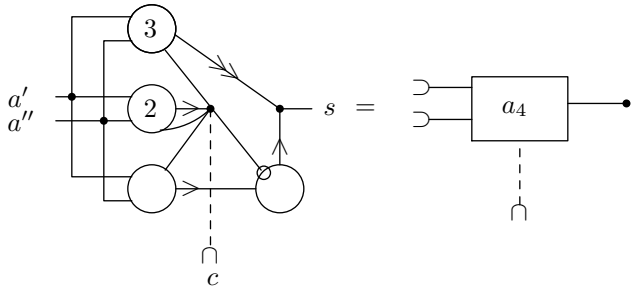


in terms of elementary gates. The the number of arrows on the output line indicated the number of unit delays (τ) introduced by the element. The arrow notation also served to indicate the output line. Thus

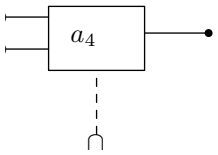


was used to indicate a construct with total delay of 2τ, where τ is the basic gate delay time.

The notion of composition was clearly established, so that, for instance, the adder circuit



was subsequently represented by



which was termed a *block symbol*.

4.2.1 Complexity

Due to the choice of purely serial and synchronous operation, it was expected that

the logic (CA and CC) would require a few hundred vacuum tubes and the memory would require around 2000, for a total count of under 2500.

5. Conclusion

The computer defined in the *First Draft Report on the EDVAC* was never built and its architecture and design seem now to be forgotten. The Report was a fundamental influence on Turing's work. However, Turing's design, the *Pilot ACE*, was only built after long delay caused by indecision on the part of the National Physical Laboratory, and long after Turing had left. Thus, even in its time, the von Neumann Report was not as influential as would have been expected.

This paper has given an indication of the nature of the design and of some of the innovations which were present in this first computer definition.

Acknowledgments

This work was supported, in part, by a Grant from Apple Computer, Inc.

References

[1] Von Neumann, J., "First Draft of a Report on the EDVAC," Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945. Corrected and complete version published in *IEEE Annals of the History of Computing,* vol. 15, No.4, 1993, pp. 27-75.

[2] Goldstine, H. H., *The Computer from Pascal to von Neumann*, Princeton University Press, 1972.

[3] Burks, A. W., "From ENIAC to the Stored-Program Computer: Two Revolutions in Computers," in *A History of Computing in the Twentieth Century*, eds. N. Metropolis, J. Howlett, and Gian-Carlo Rota, Academic Press, 1980.

[4] Von Neumann, J., "First Draft of a Report on the EDVAC," incomplete (first 5 of 15 completed Chapters), in *The Origins of Digital Computers*, ed. B. Randell, Springer-Verlag, 1973.

[5] Cohen, D., "On Holy Wars and a Plea for Peace," USC/ISI, April 1980.

[6] Knuth, D. E., "Von Neumann's First Computer Program," *Computer Surveys*, vol. 2, No. 4, Dec. 1970, pp. 247-260.

[7] Stern, N., *From ENIAC to UNIVAC, An Appraisal of the Eckert-Mauchly Computers*, Digital Press, 1981, ISBN 0-932376-14-2.

[8] Burks, A. W., Goldstine, H. H., and von Neumann, J., "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," Second Edition, 2 September 1947, Institute for Advanced Study, Princeton, N.J.

[9] Weik, M. H., "A Survey of Domestic Electronic Digital Computing Systems," Ballistic Research Laboratories Report No. 971, December 1951, Aberdeen, Md. **Correction 15 June 2006:** The date of this report is 1955, not 1951. This is confirmed in the *Third Survey of Domestic Electronic Digital Computing Systems,* Martin H. Weik, BRL Report No. 1115, March 1961.

[10] Eckert, J. P. Jr. and Mauchly, J. W., "AUTOMATIC HIGH-SPEED COMPUTING: A Progress Report on the EDVAC," Report of Work under Contract No. W-670-ORD-4926, Supplement No. 4, 30 September 1945.

[11] Aspray, W. and Burks, A., eds., *Papers of John von Neumann on Computers and Computer Theory,* Charles Babbage Institute Reprint Series for the History of Computing; v. 12, 1987, ISBN 0-262-22030-X.

[12] Turing, A. M., "Proposals for Development in the Mathematics Division of an Automatic Computing Engine (ACE)," proposal presented to the National Physical Laboratory, 1945. Reprinted as Com Sci 57, National Physical Laboratory, April 1972.

[13] Williams, M. "The Origins, Uses and Fate of the EDVAC," this issue.