The Computer as von Neumann Planned It

M.D. GODFREY

D.F. HENDRY

We describe the computer defined in von Neumann's unpublished paper "First Draft of a Report on the EDVAC," Moore School of Electrical Engineering, University of Pennsylvania, June 30, 1945. We discuss motivation for the architecture and design, and contrast the machine with the EDVAC that was actually constructed.

John von Neumann made a key contribution to the understanding and development of computer architecture and design in his unpublished report titled "First Draft of a Report on the EDVAC."[1] However, in reading work that refers to this report and to the EDVAC computer (Eckert and Mauchly say the acronym stands for Electronic Discrete VAriable Computer[2]) which it described, some perplexing observations emerge:

- The constructed EDVAC is usually described as being based on the von Neumann report.[1]
- 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, Aspray and Burks' edition of the *Papers of John von Neumann on Computers and Computer Theory,* p. xv.[3]) 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.
- The EDVAC has been referred to 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[4] indicates 40 bits, others (for example, Burks[5]) say 32 bits. The only known publication giving the correct value (44 bits) is Knuth's article.[6] The BRL Report[7] (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" described by von Neumann in the report and the "EDVAC" 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." (Eckert and Mauchly's "Automatic High-Speed Computing: A Progress Report on the EDVAC"[2] was changed to "unclassified" in 1947.) Subsequently, 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 article is to present the architecture given in the von Neumann report in a form that is accessible to a wider audience and to translate into modern terminology the formal machine definition given in the von Neumann report. We also compare this definition with 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 by Eckert and Mauchly.[2] The sections of their report that 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. Williams' article[8] in this issue reviews the actual performance of the single constructed EDVAC, which was delivered to BRL.

However, our article 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 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 stack) should be incorporated into the report. This was never done. The improvements were based on results from a sort program that 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 by the incomplete draft form of the original and the propagation of accumulating errors in the reprints that have subsequently appeared.[3,9,10] These reprints have carried over the

1058-6180/93/0100-0011$03.00 © 1993 IEEE

original errors and introduced new errors. Since one reprinting[3] was based on a previous reprint,[10] rather than on the original, further compounding of typographical errors has occurred. The incomplete copy in Randell's *Origins of Digital Computers*[9] is not very useful as it only includes the first five introductory chapters. The inaccurate copies reprinted by Aspray and Burks[3] and Stern[10] make von Neumann's original intent quite hard to discover, mainly because of numerous mistakes in the mathematical notation. A typical paragraph in Stern's reprint[10] (second paragraph, p. 239) reads as follows:

> 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 $p = 0, 1, \ldots, 31$ (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[10] (p. 242) the notation switches from μ to u, but then later (p. 243) p is switched to ρ since the typist stopped typing p and wrote in ρ by hand.

The fact that the first reprint[9] contains only the first five of 15 chapters has led to additional confusion. For example, the March 1992 issue of *Computing Reviews* contains a review of *John von Neumann and the Origins of Modern Computing* by William Aspray. 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 five chapters reprinted by Randell.[9] 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 five introductory chapters. All of the substance of the EDVAC design and architecture as expressed by von Neumann is contained in the subsequent Chapters 6 through 15.

The original manuscript from the Moore School[1] is easier to read than the published versions: The manuscript 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. The first author of this article has prepared a corrected version that reconstructs what was surely von Neumann's intended writing. This version has not been published, but it is hoped that this will be possible in the future (perhaps in a future issue of the *Annals*). The manuscript was converted 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. (For the time being, readers can obtain a copy of the report in PostScript form by anonymous FTP to isl.stanford.edu, file: pub/godfrey/reports/vnedvac.ps.)

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[11] 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 copy and this introductory guide will make this key contribution more accessible.

The two EDVACs

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[12] 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 because of 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 proofread 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 is the position as described to me by Harry Huskey.

M-EDVAC was a serial, synchronous, 44-bit word, four-address (three operand addresses and the next instruction address), binary machine with 12 operation codes. It had four registers, but these do not appear to have been addressable. It used parallel comparison of the two arithmetic units for error checking. (See Williams' article in this issue[8] for details about the realized reliability and performance of the machine.) These and other features of the machine are summarized in Table 1, an excerpt from a Ballistic Research Laboratories report.[7] This table was updated in BRL Report 1010,[13] which superseded the earlier report. However, the only important change was the inclusion of floating-point arithmetic, which was a late addition to the machine. Estimates of the number of components were also increased, probably reflecting the added floating-point circuits. Floating-point performance was not stated.

Table 1. M-EDVAC (specifications from Martin Weik, BRL Report No. 971[7]).

Manufacturer

Moore School of Electrical Engineering, University of Pennsylvania

Operating agency

US 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 µs (min 192, max 1,536)
Multiply time (including storage access): 2,880 µs (min 2,208, max 3,552)
Divide time (including storage access): 2,930 µs (min 2,256, max 3,600)
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	µs Access
Mercury acoustic delay line	1,024	48-384
Magnetic drum	4,608	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	sexadecimal 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.

Number of circuit elements

Tubes:	3,563	
Tube types:	19	
Crystal diodes:	8,000	
Magnetic elements:	1,325	(relays, coils, and trans.)
Capacitors:	5,500	approx.
Resistors:	12,000	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 conditioner	25 kW
Space occupied: air conditioner	6 sq. ft.
Total weight: air conditioner	4,345 lbs.
Capacity: air conditioner	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

(Table 1 is continued on the following page.)

Table 1. M-EDVAC (continued).

Personnel requirements

Daily operation: Three 8-hour shifts. No. of technicians: 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 1955) 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 four addresses of instruction to be executed by computer.
Tape-reader error detection.

vN-EDVAC was a serial, synchronous, 32-bit word, zero-address, binary machine with a hierarchical operation-code structure of eight basic codes, 10 subcodes, and one modifier. It had three nonaddressable registers, organized as a *stack* mechanism. Tagged memory was used to distinguish instructions from data. This feature is more fully explained in the later subsection "Instruction definition."

Table 2 (on page 16) compares the main features of the two designs.

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.

We can 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.

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). (See the glossary on page 15 for a key to acronyms and other terms.) Address modification is carried out by computing 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 [μ, ρ], but this is purely for design reasons. All addresses are 13-bit word addresses.

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 twos complement form. Thus, the range of standard numbers is $-1 \leq n < 1$ with a precision of approximately eight decimal digits. At this point and throughout the report, despite a couple of switches of notation, von Neumann is clearly a "little-endian."[14] 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.

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 the CA. O_{ca} receives the output of operations which use I_{ca} and J_{ca} as inputs. It always acts as an accumulator; that is, all results are formed by

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

However, the store operations optionally allow clearing of O_{ca} after the store operation. All store operations store O_{ca}. Figure 1 shows the interconnection of these registers. (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 this, 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.
×	Clear accumulator
x_1	Implicit load immediate x_1
y_1	Implicit load immediate y_1
×	Multiply and accumulate
x_2	
y_2	Repeat
×	for
x_3	
y_3	remaining data
×	
x_4	
y_4	
×	
$\rightarrow M[addr(S)]$	Store result at address of S

A more fully parameterized 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.

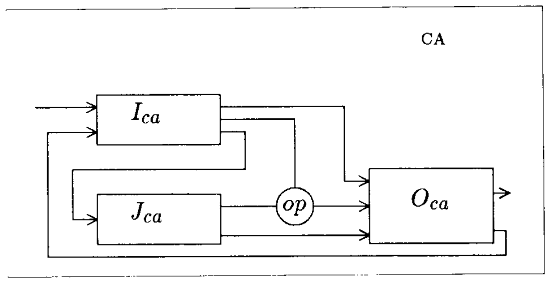


Figure 1. Interconnection of registers I_{ca} and J_{ca}.

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.

The CC section is based on a conventional instruction-sequencing mechanism for normal instruction processing. Given that the address in the PC points to the current instruction in memory, the instruction at that address is fetched, decoded, and executed; the PC is incremented; and the operation cycle is repeated. (Note that M-EDVAC loaded the next PC value from a field in the current instruc-

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
$\boxed{M_1}$	1-bit memory
$\boxed{\text{lk}}$	k-bit (serial) memory
minor cycle	32-bit word
major cycle	32 words of memory

Table 2. Von Neumann's EDVAC (vN-EDVAC) compared with the Moore School EDVAC (M-EDVAC).

	Basic design		
	vN-EDVAC	**M-EDVAC**	
Timing	Synchronous	Synchronous	
Operation	Sequential	Sequential	
	Numerical system		
	vN-EDVAC	**M-EDVAC**	
Internal number system	Binary	Binary	
Binary digits per word	32	44	
Data bits per word	31	32	
Memory tag bits	1	0	
Bits per 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 (nonaddressable)	4 (nonaddressable (?))	
Number range	$-(-2^{-30}) \leq x \leq (1-2^{-30})$	$-(-2^{-43}) \leq x \leq (1-2^{-43})$	
	Storage		
	vN-EDVAC	**M-EDVAC**	
Media	Words	Words	μs Access
Mercury acoustic delay line	8,192	1,024	48-384
Magnetic drum		4,608	17,000
	Number of circuit elements		
	vN-EDVAC	**M-EDVAC**	
Tubes	2,000-3,000 (est.)	3,563	
Tube types		19	
Crystal diodes		8,000	
Magnetic elements		1,325	(relays, coils, and trans.)
Capacitors		5,500	approx.
Resistors		12,000	approx.
Neons		320	approx.

tion word, as is common in many microcode systems.) There are two exceptions to this standard instruction-processing loop. The first is a control transfer instruction that loads the 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 the PC are loaded into I_{ca}, and the PC is incremented in the normal way.

In the report, the operations within the 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 3 and 4 summarize the notation and meaning for the unpooled orders. The actual instructions are termed the *pooled* orders and are summarized in Table 5. As can be seen, there are eight instruction types (t'). The CA instructions use the subcode 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 4 with the pooled orders in Table 5, 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 with 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 5 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 $i_1 \ldots i_{30}$ 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.

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 5) loads the word that 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 to achieve some of the effects (and side effects) of the delayed branch (RISC) operation.

Table 6 is copied directly from the von Neumann report.[1] It shows the manner in which von Neumann summarized the logical definition of the machine.

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 with 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.

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 1 MHz the circulation time of the delay line was about 1,000 μs. The size of the delay lines

Table 3. Notation for instruction fields.

i_0	Tag bit
	0 data
	1 instruction
t	Unpooled instruction code
t'	Pooled instruction code
w	Operation subcode 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

Table 4. Operation code definitions.

Unpooled types (t)

Order	Name	Definition
α	CC operations (stack)	See α operations 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

α operations

Modifier	Value	Operation	Definition
$w =$	0	+	$(I_{ca} + J_{ca}) + O_{ca} \rightarrow O_{ca}$
	1	−	$(I_{ca} - J_{ca}) + O_{ca} \rightarrow O_{ca}$
	2	×	$(I_{ca} \times J_{ca}) + O_{ca} \rightarrow O_{ca}$
	3	/	$(I_{ca} / J_{ca}) + O_{ca} \rightarrow O_{ca}$
	4	√	$\sqrt{I_{ca}} + O_{ca} \rightarrow O_{ca}$
	5	i	$I_{ca} \rightarrow O_{ca}$
	6	j	$J_{ca} \rightarrow O_{ca}$
	7	s	perform i or j depending on sign of O_{ca}
	8	db	decimal → binary
	9	bd	binary → decimal

Memory addressing

$\mu[\mu_7 \ldots \mu_0]$	Major memory cycle (segment)
$\rho[\rho_4 \ldots \rho_0]$	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 5. Pooled orders.

Type (t')	instruction format (least significant bit at right, bit width of each field indicated below each instruction)	Definition	Meaning
$i_0 = 0$	fields: i_0; values: [| 0]; widths: 1	$M[PC] \rightarrow I_{ca}$	load immediate
γ	fields: t' i_0; values: [| 0 | 1]; widths: 3 1	$M[PC+1] \rightarrow I_{ca}$	load immediate
$\alpha + \delta$	fields: μ ρ w c t' i_0; values: [| | | | | 1 | 0]; widths: 8 5 10 4 1 3 1	OP; $O_{ca} \rightarrow M[\mu, \rho]$	stack operation; store
$\alpha + \epsilon$	fields: w c t' i_0; values: [| | | 2 | 1]; widths: 23 4 1 3 1	OP; $O_{ca} \rightarrow M[PC+1]$	stack operation; store immediate
$\alpha + \theta$	fields: w c t' i_0; values: [| | | 3 | 1]; widths: 23 4 1 3 1	OP; $O_{ca} \rightarrow I_{ca}$	stack operation; load
α	fields: w c t' i_0; values: [| | | 4 | 1]; widths: 23 4 1 3 1	OP	stack operation
β	fields: μ ρ t' i_0; values: [| | | 5 | 1]; widths: 8 5 15 3 1	$M[\mu, \rho] \rightarrow I_{ca}$	load
ζ	fields: μ ρ t' i_0; values: [| | | 6 | 1]; widths: 8 5 15 3 1	$M[\mu, \rho] \rightarrow PC$	control transfer
η	fields: not defined t' i_0; values: [| 7 | 1]; widths: 3 1	I/O	input-output

Note: Von Neumann did not assign numeric codes to the eight pooled orders. (At the end of the report he indicated that he would do that next.) We 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 3, and w is the CA order modifier as defined in Table 4.

was determined from the fact that 1 ms 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 remarked, "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.

E-elements and logic. This is, as far as we are aware, the first substantial work (since Babbage) that 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 Eckert and Mauchly's "Progress Report on the EDVAC"[2] and Burks, Goldstine, and von Neumann's "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument."[12]

Table 6. Instruction definition.

(I) Type	(II) Meaning	(III) Short Symbol	(IV) Code Symbol
			Minor cycle $I = (i_v) = (i_0 i_1 i_2 \cdots i_{31})$
Standard Number or Order (γ)	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 $\rightarrow$ A or wh $\rightarrow$ A.	Nξ	$i_0 = 0$
Order $(\alpha)+(\delta)$	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:	w $\rightarrow \mu\rho$ or wh $\rightarrow \mu\rho$	$i_0 = 1$

w.decimal	w.binary	w	w.decimal	w.binary	w
0	0000	+	5	0101	i
1	0001	−	6	0110	j
2	0010	$\times$	7	0111	s
3	0011	$\div$	8	1000	db
4	0100	$\sqrt{\ }$	9	1001	bd

Order $(\alpha)+(\epsilon)$		w $\rightarrow$ f or wh $\rightarrow$ f	
Order $(\alpha)+(\theta)$		w $\rightarrow$ A or wh $\rightarrow$ A	
Order (α)	h means that the result is to be held in O_{ca}. $\rightarrow \mu\rho$ means that the result is to be transferred into the minor cycle ρ in the major cycle μ; $\rightarrow$ f, that it is to be transferred into the minor cycle immediately following upon the order ϵ; $\rightarrow$ A, that it is to be transferred into I_{ca}; no $\rightarrow$, that no disposal is wanted (apart from h).	wh	
Order (β)	Order to transfer the number in the minor cycle ρ in the major cycle μ into I_{ca}.	A $\leftarrow \mu\rho$	
Order (ζ)	Order to connect CC with the minor cycle ρ in the major cycle μ.	C $\leftarrow \mu\rho$	

The simplest gate was drawn as shown in Figure 2a, while an inverter was drawn as in Figure 2b. A two-input OR gate would be represented as in Figure 2c. The notation shown in Figure 2d was used for a two-input AND gate. A three-input AND gate was given as in Figure 2e. Thus, the number inside the circle indicated the minimum number of active inputs required to drive the output active. A two-input AND gate would be defined as shown in Figure 2f in terms of elementary gates. 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 the notation in Figure 2g indicated a

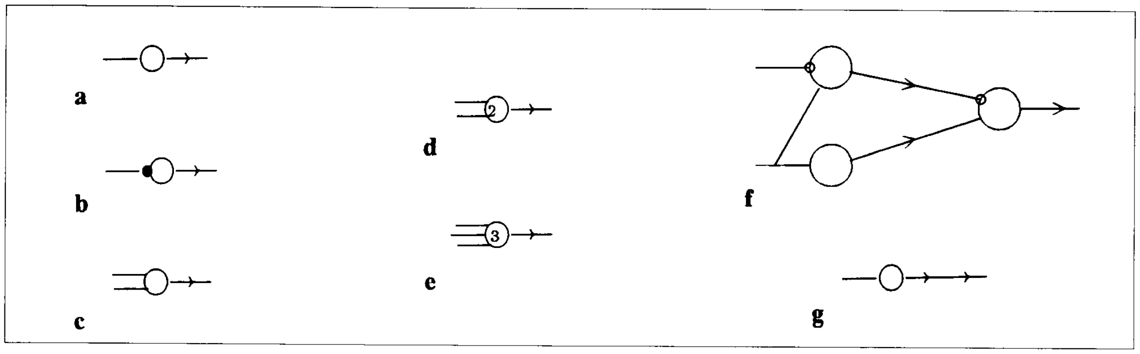


Figure 2. Notation used in the von Neumann first draft: (a) simplest gate, (b) inverter, (c) two-input OR gate, (d) two-input AND gate, (e) three-input AND gate, (f) two-input AND gate defined in terms of elementary gates (arrows on the output line indicate the number of unit delays introduced by the element), (g) a construct with total delay of 2τ, where τ is the basic gate delay time.

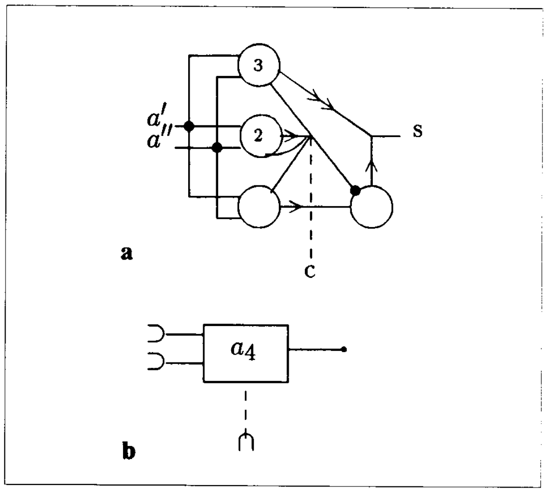


Figure 3. An adder circuit (a) would be represented as a block symbol (b).

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 shown in Figure 3a was subsequently represented as in Figure 3b. This notation was termed a *block symbol.*

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 2,000, for a total count of under 2,500.

The computer defined in the "First Draft of a 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 built only 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 article has given an indication of the nature of the design and of some of the innovations present in this first computer definition. ■

Acknowledgment

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

References

1. J. von Neumann, "First Draft of a Report on the EDVAC," Moore School of Electrical Engineering, Univ. of Pennsylvania, Philadelphia, June 30, 1945.
2. J.P. Eckert, Jr., and J.W. Mauchly, "Automatic High-Speed Computing: A Progress Report on the EDVAC," Report of Work under Contract No. W-670-ORD-4926, Supplement No. 4, Moore School Library, Univ. of Pennsylvania, Philadelphia, September 30, 1945.
3. W. Aspray and A. Burks, eds., *Papers of John von Neumann on Computers and Computer Theory,* Charles Babbage Inst. Reprint Series for the History of Computing, Vol. 12, MIT Press, Cambridge, Mass., and Tomash Publishers, Los Angeles, 1987.
4. H.H. Goldstine, *The Computer from Pascal to von Neumann,* Princeton Univ. Press, Princeton, N.J., 1972.
5. A.W. Burks, "From ENIAC to the Stored-Program Computer: Two Revolutions in Computers," in *A History of Computing in the Twentieth Century,* N. Metropolis, J. Howlett, and Gian-Carlo Rota, eds., Academic Press, New York, 1980, pp. 311-344.
6. D.E. Knuth, "Von Neumann's First Computer Program," *Computer Surveys,* Vol. 2, No. 4, Dec. 1970, pp. 247-260.

7. M.H. Weik, "A Survey of Domestic Electronic Digital Computing Systems," Report No. 971, Ballistic Research Laboratories, Aberdeen, Md., 1955.

8. M.R. Williams, "The Origins, Uses, and Fate of the EDVAC," *IEEE Annals of the History of Computing*, Vol. 15, No. 1, Jan. 1993, pp. 22-38.

9. J. von Neumann, "First Draft of a Report on the EDVAC," incomplete (first five of 15 completed chapters), in *The Origins of Digital Computers*, B. Randell, ed., Springer-Verlag, Berlin, 1973, pp. 355-364.

10. N. Stern, *From ENIAC to UNIVAC, An Appraisal of the Eckert-Mauchly Computers*, Digital Press, Bedford, Mass., 1981.

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

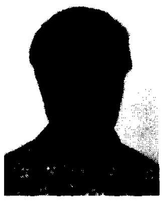
12. A.W. Burks, H.H. Goldstine, and J. von Neumann, "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," 2nd edition, Institute for Advanced Study, Princeton, N.J., 1947.

13. M.H. Weik, "A Second Survey of Domestic Electronic Digital Computing Systems," Report No. 1010, Ballistic Research Laboratories, Aberdeen, Md., 1957.

14. D. Cohen, "On Holy Wars and a Plea for Peace," Univ. of Southern California/ISI, Los Angeles, 1980.

Michael D. Godfrey is a visiting professor in electrical engineering and statistics at Stanford University. He has been a member of the statistics faculty at Princeton University and a member of the technical staff at Bell Laboratories, Murray Hill, N.J. He has also held the Schlumberger chair at the Imperial College of Science, Technology and Medicine in London, and worked as director of research at Sperry Univac. Throughout his career he has had a strong interest in computing and its history.

Godfrey obtained a BSc from the California Institute of Technology in 1959 and a PhD from the University of London in 1962.



David F. Hendry is a founder of T-H Engineering, Inc., in Altadena, Calif. He has held positions in the Silicon Structures Program at Caltech and at the Jet Propulsion Laboratory. He has also been a faculty member at the Polytechnic of the South Bank, London, and at the University of London Institute of Computer Science. His research interests are in software technology and computer design, particularly with respect to VLSI implementation.

Hendry completed his BSc in mechanical engineering at the University of Glasgow in 1959 and a postgraduate diploma in industrial administration in 1960.

Godfrey can be reached at the Durand Building, ISL, Electrical Engineering Department, Stanford University, Stanford, CA 94305.