THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

FROM ENIAC TO THE STORED PROGRAM COMPUTER: TWO REVOLUTIONS IN COMPUTERS

Arthur W. Burks

January 1978

Logic of Computers Group
Technical Report No. 210

with assistance from

National Science Foundation
Grant No. MCS76-04297
Washington, D.C.

ensgn
UMR0866

FROM ENIAC TO THE STORED PROGRAM COMPUTER: TWO REVOLUTIONS IN COMPUTERS

Arthur W. Burks

1. Introduction 1
2. ENIAC Hardware and Arithmetic Design 2
3. ENIAC Organization and the Differential Analyzer 8
4. Programming ENIAC 12
5. Evaluation of ENIAC 18
6. High-Speed Read-Write Electronic Stores 20
7. Code and Organization for the Stored Program Computer 22
8. Conclusion 28
Figures 1-20 29

This is a revised and expanded version of my paper at the International Research Conference on the History of Computing, Los Alamos Scientific Laboratory, Los Alamos, New Mexico. It is to be included in the proceedings of the conference, to be edited by Jack Howlett and published by Academic Press.

The paper was written under NSF Grant No. MCS76-04297. I wish to thank Brian Randall and Cuthbert Hurd for helpful comments.

---

1. Introduction

There has been a long controversy as to who invented the stored program computer. Unfortunately, this question is overly simplistic. The development of the stored program computer took place in many steps and involved many people. I will trace this development through its main stages, starting with the antecedents of the ENIAC and ending with the first generation of stored program computers: EDVAC, IAS, Whirlwind, EDSAC, UNIVAC I, and many others.

The period from 1935 to 1952 saw two important revolutions in computers. The first was the employment of vacuum tubes to make a fast, reliable, powerful, general-purpose computer. This development began with John Atanasoff's slow, special-purpose electronic computer. It culminated in ENIAC, shown in Figure 1. ENIAC was developed, designed, and built by a group of engineers, including myself, under the direction of Presper Eckert and John Mauchly. ENIAC was revolutionary: it was the first electronic, digital, general-purpose, scientific computer, and it computed 1000 times as fast as its electromechanical competitors.

The second revolution was the stored program computer. It too had important antecedents, which I'll explain in due course. There were two main steps. Eckert and Mauchly invented the circulating mercury delay line store, with enough capacity to store program information as well as data. John von Neumann created the first modern order code and worked out the logical design of an electronic computer to execute it.

This early work on electronic computers arose out of the background of modern electronics and out of a well-established technology of mechanical and electromechanical computing. See Figure 2. Many of the circuits of ENIAC were electronic versions of mechanical and electromechanical digital computing devices, and the architecture of ENIAC derived from the mechanical differential analyzer. Thus ENIAC established electronic technology as the way to do computing hitherto done mechanically and electromechanically.

ENIAC necessarily used electromechanical equipment for input and output, and thereby tested the relative merits of the two technologies for computing. Not surprisingly, some of the older and more experienced engineers at the Moore School were sceptical about ENIAC. "18,000 vacuum tubes? No one has ever operated a system of more than 100 tubes or so. At any moment at least one tube will be inoperative, and that may spoil the answer!" But the electromechanical I-O of ENIAC and EDVAC gave at least as much trouble as the elec-

tronics. And so it has remained to this day.

In my discussion I will largely ignore input-output equipment, for the hardware revolutions of ENIAC and the first stored program computers were based on dramatic changes in the nature, speed, and cost of internal components. The computing part of ENIAC dominated its I-O equipment, both in size and cost. Today the situation is reversed. The logical equivalent of internal ENIAC can be put on the end of your finger, and it costs much less than ENIAC equivalent I-O.

2. ENIAC Hardware and Arithmetic Design

Vacuum tubes had been developed for radio and telephone communication. In the mid-1930's they were still used mostly in analog or continuous fashion, though flip-flops, counters, and small vacuum-tube switching circuits had been invented and were used occasionally.

John Atanasoff of Iowa State College pioneered in using vacuum tube circuits for computing. Starting about 1935 he designed and partially built a special-purpose digital electronic computer for solving simultaneous equations by Gaussian elimination. He was assisted by Clifford Berry. Atanasoff and Berry stored binary numbers on capacitors embedded in rotating drums, and they did arithmetic and logic with vacuum tube circuits. These circuits were slow, operating at 60 pulses per second, which was about the speed of relay circuits.

In the spring of 1941 Atanasoff conceived of a digital electronic version of the differential analyzer. He communicated his idea to John Mauchly, who was to join the faculty of the University of Pennsylvania's Moore School of Electrical Engineering in the fall of 1941. The Moore School had a differential analyzer, which was used some but often stood idle.

After the USA entered World War II, the Moore School became a center for calculating firing tables. The differential analyzer was soon used full-time for calculating trajectories. A group of young ladies also calculated trajectories, using electrically powered mechanical calculators: Friedens, Monroes, Marchants. Herman Goldstine was the Army Lieutenant in charge of this activity.

One day Mauchly suggested to Goldstine that trajectories could be calculated much faster with vacuum tubes. Herman thought John's suggestion a good one, and asked for a proposal. John and Pres Eckert then wrote "Report on an Electronic Diff.* (sic) Analyzer" (April 2, 1943). This was submitted to the U.S. Army's Ballistic Research Laboratory, Aberdeen Proving Grounds, on behalf of the Moore School.

In this proposal Eckert and Mauchly offered a machine that could compute ballistic trajectories at least 10 times as fast as the differential analyzer, and at least 100 times as fast as a human computer with a desk calculator. Goldstine persuaded the Ballistic Research Laboratory to fund the proposal. The "electronic diff.* analyzer" later became the "Electronic Numerical Integrator and Computer," or "ENIAC" for short.

John and Pres proposed to achieve this very high computing speed by operating vacuum tube circuits at 100,000 pulses per second. Physicists had used fast electronic circuits for counting cosmic rays, but these did not need to be, and were not, very reliable. The first development task on the ENIAC project was to design reliable counters that worked at 100,000 pulses per second and to show by test that switching circuits could work at a comparable speed. The final ENIAC operated at 100,000 pulses per second and thus became the first computer to fully exploit the vacuum tube technology of the time.

We will discuss some of ENIAC's arithmetic circuits in a moment, but to put this discussion in perspective we must first make a few general remarks about the machine's organization. ENAIC was composed of a large number of semiautonomous computer units, arranged as in Figure 3. There were 25 computing units proper: 20 accumulators, one high-speed multiplier, 3 function table units, and a divider & square-rooter. There was a constant transmitter for input and a printer unit for output. Each of these units had its own local program circuits, and there was also a central program control, called the master programmer. All these units were timed by signals from a central clock, the cycling unit.

The programming circuits of ENIAC were very important and occupied a large part of the machine. In the computing units proper the control circuits were about equivalent in size to the arithmetic circuits, and more complex. We will discuss ENIAC programming in detail later (sec. 4). For the moment it suffices to say that subroutines were set up mechanically on local program controls. These subroutines were connected to the master programmer, which operated them in proper sequence, each subroutine being employed a fixed number of times or until a branch occurred.

One last point needs to be made before we return to ENIAC arithmetic. The units of ENIAC were not fully autonomous, because computing always required the cooperation of two or more of them. For example, addition was accomplished by transmitting from one accumulator to another. Correspondingly, local program controls on these accumulators had to be stimulated simultaneously.

The arithmetic design of ENIAC was influenced mainly by two kinds of calculators: mechanical desk calculators, electrically-powered and hand-operated; and electromechanical card-operated IBM machines. We knew of the relay computers built and being built at the Bell Telephone Laboratories by George Stibitz, Sam Williams, and others. We also knew of Harvard Mark I, a general-purpose electromechanical calculator designed by Howard Aiken and several IBM engineers. We did not know of the work of Conrad Zuse or of the British Colossi machines.

In mechanical technology, arithmetic registers were formed of toothed wheels, and carrying was done with cams and ratchets. Numbers were transmitted by shafts with gears, or the linear equivalent, toothed racks. In electromechanical technology, relays were used for switching and stepping switches were used for counting. Relays could be "locked" to form flip-flops. Numbers were transmitted as electrical pulses on wires. IBM machines and Harvard Mark I combined mechanical and electromechanical technology.

In ENIAC, the toothed wheel and stepping switch became the electronic counter. A decade ring counter is shown in Figure 4. Switching was done with vacuum tubes. The 6SA7 (tube 9) of Figure 5 functions as a "gate" or switch. The program control is activated when a program pulse (from another program control or the master programmer) sets the flip-flop (tubes 1, 2, 3, and 4). When the flip-flop is set, it activates one input of the 6SA7 gate. The next program pulse from the cycling unit is then passed by the 6SA7. This pulse rests the flip-flop, closing the 6SA7 gate, and via tubes 10 and 11 drives a program line to start the program controls used in the next operation.

Both numbers and control signals were transmitted as voltage pulses. We called these "pulses" when they lasted about 3 microseconds, and "gates" when they lasted much longer. The basic pulses and gates came ultimately from the cycling unit (see fig. 13 below).

We have listed the main electronic building blocks of ENIAC: counters, flip-flops, and vacuum tube switches. Let us now move up a full level in the hierarchy of ENIAC structure and consider how the high-speed multiplier worked. For brevity I'll call it the "multiplier," and the two numbers to be multiplied the "ier" and the "icand".

Since the multiplier could not multiply by itself, we need to consider it and four associated accumulators: ier accumulator, icand accumulator, and two partial products accumulators. The three uncovered panels in the center of Figure 6 constitute the multiplier, and the associated accumulators are to

the left and right.

The time required for an ENIAC multiplication depended only on the number of ier digits called for in programming a multiplication. After a preparatory addition time, ENIAC multiplied one digit of the ier by the whole icand in a single addition time, dividing the partial product into left-hand (LH) and right-hand (RH) components. It did this by means of a pre-wired multiplication table. Two final addition times were used for combining the LH and RH partial products and making complement connections.

For the reader interested in the task of an ENIAC design engineer, I'll explain how the multiplication circuits worked. Figures 7 and 8 show a sample of these circuits. The figures cover a two-digit ier, a two-digit icand, and that part of the multiplication table and its output tubes needed for the icand digits of 0, 1, 2, and 3.

The ier selector in the upper left-hand corner of Figure 7 consists of and-gates with two inputs (tubes A0-A9 and B0-B9). Each selector gate is connected on one input to the ier accumulator and on its other input to a control wire. These control wires select the digit positions in turn; line 1 for the units digit, then line 2 for the tens digit. The selected ier selector gate activates the multiplication table through a driver (P, Q, R, S, etc.).

The multiplication table proper is a resistor matrix, each resistor establishing a direct electrical connection from an input digit wire (0-9) to a table output gate below (tubes C1, D1, D2', E1-E4, F2-F4, and G1-G4). A multiplication table input wire which is activated drops its voltage, thereby closing those output gates to which it is directly connected via a resistor.

The table output gates also receive groups of pulses from the cycling unit; see Figure 13 below. Each gate left on by the multiplication table passes these pulses, which go through tubes U, V, etc. to the icand selectors below. Thus the table output gates produce, on separate wires, pulses which represent the product of the selected ier digit by all the digits 1 through 9, divided into LH and RH components. The LH components go to the LH icand selector (H2-H9, J2-J9), then to the LH shifter (M1-M4), and finally to the LH partial products accumulator. The right-hand components go to the RH ier selector (K1-K9, L1-L9), RH shifter (N1-N4), and RH partial products accumulator.

We will trace the logical behavior of these circuits for a specific example. Let the ier be 40 and the icand 30, so the desired product is 1200, consisting of a LH component of 1000 and a RH component of 200. Consider the action of the multiplier while it processes the tens ier digit, which is 4.

The program circuits hold line 2 active while the tens digit of the ier is being processed. Hence the ier selector gates A0-A9 are activated from the right. (The shifter gates M3, M4, and N3, N4 are also activated by control line 2.) Tube A4 is also activated on the left by the ier accumulator, so it goes on, turning P off and Q on, thereby lowering the voltage on input line 4. This line turns off all the table output gates to which it is directly connected through resistors, leaving the remaining gates on. As far as the icand digit 3 is concerned, the LH gate D1 is on and D2' is off, while the RH gate G2 is on and gates G1, G2', G4' are off. Gate D1 sends one pulse to row 3 of the LH icand selector (H3, J3) and gate G2 sends two pulses to row 3 of the RH ier selector.

These streams of pulses are gated by the icand selectors and then shifted by the shifters before being transmitted to the partial products accumulators. Since the icand is 30, a pulse representation of 100 comes from the LH selector and a pulse representation of 20 from the RH selector. Because line 2 is activated by the program circuits these are shifted one position to the left, so 1000 goes to the LH partial products accumulator and 200 to the RH partial products accumulator. At the end of the multiplication process these would be combined to give 1200, which is the product of 40 times 30.

This completes our explanation of how the ENIAC multiplier and its associated accumulators performed high-speed multiplication. The multiplication table was invented for mechanical calculators about 100 years ago. It was used in the IBM 601 crossfooting multiplier of 1931, which could be programmed to compute things like $AxB+C+D$, $A+C-D$, and $AxB-C$. This machine will be described below in connection with ENIAC programming (sec. 4 and figs. 14 and 15).

Was this particular adaptation from mechanical technology wise for ENIAC? Fast multiplication made mechanical and electromechanical calculators superior to their predecessors. But ENIAC did many more operations, solved much more complicated problems, and had to be slowly programmed for each of these. Did fast multiplication in ENIAC provide a sufficient gain in speed over the method of repeated addition, or the faster method of repeated addition-or-subtraction, to justify it? Would the binary system have been a better choice than the decimal system? These are interesting questions which I cannot go into here.

I'll tell a story about the IBM 601 and ENIAC, which illustrates a general point about invention. An inventive IBM engineer wrote me a few years ago concerning the ENIAC multiplier: "I felt you were doing electronically almost exactly what we had had in general use for 12 years in the 600 and 601." I am

reminded of what Sir John Fleming, the inventor of the vacuum tube diode, is supposed to have said of Lee de Forest's invention of the triode: "That is no invention; all de Forest did was add a third wire to my valve!"

Actually Fleming's tube was "only" a rectifier, whereas de Forest's triode was an amplifier, and amplification is what made the radio and electronics industry possible. Likewise, the computer industry became truly important when it moved from mechanical and electromechanical technology to electronics. This technological shift produced the transition from the industrial revolution to our current computer-control revolution.

Besides Atanasoff, we knew of two others who had worked on special-purpose electronic computers, in both cases for fire control. Perry Crawford wrote his master's thesis at MIT on controlling anti-aircraft guns electronically, and Jan Rajchman did preliminary development work on an electronic fire control calculator at RCA. Both developed electronic computing circuits and both invented a hardware constituent used in ENIAC, the resistor matrix function table. I described the fixed resistor table of the multiplier a moment ago. In it, a given input line drove a fixed set of output lines through resistors. One could make a variable table by installing a ten-position switch for each digit entry of the table. The resistor from the input line could then be manually switched to any one of nine output wires, representing the digits 1 through 9. ENIAC had three function table units for storing arbitrary functions, and each of these was based on a variable resistor matrix table set by switches.

The straightforward way to build a switching network, such as a multiplication table or function table, is to connect a diode between an input wire and each output wire it is to drive. The ratio of the backward resistance to the forward resistance of the diode is very high. But solid-state diodes did not exist then, a vacuum diode was about as "expensive" as a triode, so in many applications this method of switching was not practical.

The resistor matrix function table was a way of using linear resistors as switching devices in read-only memories. When Jan Rajchman told von Neumann of his invention, Johnny replied: "That can't work, Jan, it's just one big short circuit!" His point was that resistors are not rectifiers, and hence every output is connected to every input. This is clear from Figures 7 and 8. But with careful design the desired paths had much less resistance than undesired paths, and output gates could safely sense the difference. Thus in Figure 8, tube Q (line 4) does not directly drive the input to gate D1, but only through highly resistant back circuits.

The ENIAC project was funded to facilitate the preparation of firing tables. The calculation of trajectories was a good problem for computing machines, because the calculation depends critically on the drag function $G(v^2)$, which gives the resistance of the air to the movement of the shell as a function of the velocity squared. This function is an ill-behaved function of the shell's velocity, especially as the shell passes through the sound barrier. In hand calculation the resistance was read from printed tables, and on the differential analyzer the resistance was fed in from an input table. Such a table is shown on the right in Figure 9. Originally it was operated by hand. Later it was operated automatically by means of a photocell which read a black-white boundary and signaled a servomechanism feedback circuit.

All of these table input methods were too slow for ENIAC. Hence we used variable resistor matrixes set by hand switches. There were three function table units. Each had two panels of electronic equipment and a portable function table matrix box; see Figure 3. One portable matrix appears in the foreground of Figure 1 and the other two in the right rear, with operators setting switches.

A function table stored 104 entries, each with 12 decimal digits and two signs. Two signs were provided so that two numerical functions of 6 digit accuracy could be stored in one table. Two decimal digits were used to select a function value in the range 0 to 99; the extra four entries (-2, -1, 100, 101) were there to facilitate interpolation. Note that the total digit capacity of the three read-only tables was about 4000 decimal digits, or 20 times that of the 20 read-write accumulators!

When ENIAC computed a trajectory the ballistic drag function was stored in one of these function tables. Without the resistor matrix the values of this function would have had to be brought in from the outside as needed, a much slower process. At its dedication the ENIAC computed a trajectory in 20 seconds, faster than the shell itself, which took 30 seconds to reach its target! Thus the read-only resistor matrix played an essential role in the success of ENIAC.

3. ENIAC Organization and the Differential Analyzer

I have already mentioned that ENIAC was composed of 25 computing units (accumulators, multipliers, function tables, and divider & square-rooter), an input unit, and an output unit. To program ENIAC to solve a particular prob-

lem one had to interconnect the digit circuits of these units and also their programming circuits. In addition, the programming circuits had to be connected to the master programmer.

ENIAC's organization came from, and closely paralleled, that of the differential analyzer. The 1943 ENIAC proposal, on which Ordnance staked about $100,000, was titled "Report on an Electronic Diff.* Analyzer," with the following footnote explanation of the term "Diff.":

> *The word Diff. is deliberately abbreviated. Present differential analyzers operate on the basis of integrating continuously, i.e., by differential increments; the electronic analyzer, although it is believed that it would be both speedier and more accurate, would operate using extremely small but finite differences. The abbreviation "diff." may thus be considered to represent "difference" rather than "differential" in the case of the electronic device.

The differential analyzer was a mechanical analog computer for solving differential equations. It had been invented by Vannevar Bush at MIT. The Moore School had built one for itself and a similar one for Aberdeen Proving Ground. Figure 9 shows the Aberdeen machine. The differential analyzer could solve a variety of differential and integral equations, and hence was to some degree a general-purpose machine.

There were three fundamental types of calculating units in the differential analyzer: integrators for integration of functions, differential gears for addition-subtraction, and fixed gears for multiplication-division by a constant. There was no primitive unit for multiplication. Multiplication was accomplished through integration by parts; this required two integrators and one differential gear.

Inputs were inserted as initial conditions, or supplied continuously from an input table. The outputs were plotted on output tables or printed. In Figure 9 the integrators are on the left, the input-output plotting tables on the right, and the printing mechanism is in the foreground. The main bay holds the shafts used for interconnections. Differential and fixed gears were mounted on these shafts.

An integrator consisted of a metal wheel riding on a rotating glass disk at a variable distance $f(y)$ from the disk's center. For each increment dy of rotation of the glass disk, the metal wheel rotated the amount $f(y)dy$. The wheel output was thus $\int f(y)dy$. This rotational output went to a torque amplifier and thence to a shaft in the bay. Originally the torque amplifiers were mechanical, but during the war they were replaced by an arrangement of two polaroid disks, a light beam, a photocell, and a servomotor.

To set up the differential analyzer to solve a particular system of differential equations, the operator interconnected the units by shafts and gears into a pattern corresponding to the equations. The whole system was driven by a shaft representing the independent variable. This shaft was rotated by an electric motor.

Figure 10 is a diagram from Douglas Hartree's *Calculating Instruments and Machines* showing the set-up for the following integral equation, where x is the independent variable and y the dependent variable:

$$\frac{dy}{dx} = -\int \frac{dy}{dx}\, dy + \int y \cos(x+y)\, dx.$$

Scale factors and signs are omitted from the diagram. The independent variable shaft x is driven by a motor. The output was recorded from the dependent variable shaft y.

With this information about the differential analyzer as background, let us look at the structure of ENIAC. Figure 11 is from the 1943 ENIAC proposal, and shows how the units of the machine were to be interconnected and controlled so as to solve the differential equation

$$\frac{d^2y}{dt^2} = ky.$$

The solution for y is, of course, an exponential, sine, or cosine, with suitable constants.

The corresponding difference equation is $\Delta^2 y = k(\Delta t)^2 y$. For the nth step of the integration the "electronic diff. analyzer" needs to compute

$$(\Delta^2 y)_n = [k(\Delta t)^2] \times y_{n-1}$$
$$(\Delta y)_n = (\Delta y)_{n-1} + (\Delta^2 y)_n$$
$$y_n = y_{n-1} + (\Delta y)_n .$$

The constant transmitter holds $k(\Delta t)^2$ and the accumulators hold the quantities indicated in Figure 10.

I will trace one step of the integration. The constant transmitter sends $k(\Delta t)^2$ to the multiplier and the y-accumulator sends y_{n-1}. The multiplier computes $[k(\Delta t)^2] \times y_{n-1}$, which is $(\Delta^2 y)_n$ and sends it to the $\Delta^2 y$-accumulator for temporary storage. It is added from there into the Δy-accumulator to give the new value of Δy. That is then added into the y-accumulator to give the new

value of y.

Note that the $\Delta^2 y$-accumulator was used only for temporary storage. Moreover, if k and Δt are chosen so that $k(\Delta t)^2$ is a power of ten, the electronic multiplier can be replaced by a mechanical shifter of wires. The difference equations then reduce to

$$(\Delta y)_n = (\Delta y)_{n-1} + 10^{-d} y_{n-1}$$

$$y_n = y_{n-1} + (\Delta y)_n \ .$$

These can be solved with just two accumulators, one for $(\Delta y)_n$ and one for y_n. The second transmits to the first through a wire shifter which shifts the transmitted number d positions to the right, and the first then transmits to the second, the quantities received being accumulated in both cases.

This method was used to test two ENIAC accumulators in the summer of 1944. The number in each accumulator was displayed with neon lights, and it was interesting to see the contents of the accumulators going up and down, one displaying the sine and the other displaying the cosine.

These two accumulators were connected in a pattern very similar to the interconnection pattern of integrators V and VI of Figure 10. Each of these integrators feeds the other, just as each ENIAC accumulator augmented the other. In Figure 10 both integrator V and integrator VI are fed $x+y$ as the independent variable. The output from the metal wheel of each controls the distance between the metal wheel and the center of the rotating glass disk of the other. Hence one integrator produces $\sin(x+y)$ and the other produces $\cos(x+y)$.

The shift from the analog differential analyzer to the digital ENIAC required a shift in the representation of data.

In the differential analyzer, the values of variables were represented by the positions of integrator wheels and disks, and the positions of the shafts driving and driven by the integrators. In ENIAC, numbers were held in the counters of accumulators, and transmitted in pulse form over groups of wires called digit trunks. Sometimes a digit trunk was a 11-wire cable, but usually it was composed physically of digit "trays" connected to each other and to program panels by 11-wire cables. A "tray" was 8' long, an inch thick, and wide enough to carry 11 wires shielded from each other by metal partitions. Ten wires were used for decimal digits, each digit being transmitted as a pulse sequence, and one wire was used for the sign.

In Figure 12 the program panels are in the middle and the digit trays are

above them. The trays below are used for program lines in a similar way; they will be discussed in the next section.

A caveat should be entered here. I am describing the general way in which ENIAC did things and am not attempting to be complete. For example, some numbers were transmitted as slow signals over cables. The contents of the ier and icand accumulators were transmitted to the ier and icand selectors of the multiplier in this manner.

The differential analyzer had a unique shaft, the independent variable shaft. It was driven by a motor, the speed of the motor controlling the speed of computation. In ENIAC this shaft became a ten-wire trunk carrying clock signals to ENIAC units, and the motor was replaced by a complicated clock used to synchronize ENIAC operations. This clock was called "the master pulse generator" in the proposal and "cycling unit" in the final ENIAC. Its ten outputs are shown in Figure 13. There are nine distinct groups of pulses, the individual pulses lasting about 3 microseconds. There is one long pulse of 70 microseconds, called the "carry-clear gate" and used mainly in the accumulators.

Some earlier digital machines were clocked. IBM punched card calculators were timed from the rotating mechanism that pulled the cards through the machine. Thus the significance of an electric pulse from a hole depended on the position of the hole in the card.

The shift from analog differential analyzer to digital ENIAC also required the addition of programming circuits. In the differential analyzer all computing units operated simultaneously. In ENIAC two or more units were required for an arithmetic operation, but they played different roles. For example, if the contents of accumulator 7 were to be added to those of accumulator 9, accumulator 7 transmitted and accumulator 9 received. Hence programming circuits were needed to operate the arithmetic circuits in proper parallelism and sequence. These circuits will be discussed in the next section.

4. Programming ENIAC

The differential analyzer was programmed manually, with a wrench in one hand and a gear in the other. ENIAC programming was also manual, but one plugged cables and wires, and set switches. The method of programming by plugging wires was derived from the IBM plugboard. I'll explain this in connection with the IBM 601 crossfooting multiplier; see Figures 14 and 15.

The plugboard of Figure 15 is wired to calculate $AxB+C+D$. We mentioned earlier that the IBM 601 multiplied one digit of A (the ier) by all of B (the icand) in one addition time. The partial products were placed in two counters, a left-hand counter (LHC) and a right-hand counter (RHC). The numbers C and D were placed in these counters initially, and thus appeared in the final sum. The addition of C and D to the product was called "crossfooting," from an accounting term meaning to add across rather than up and down a column of numbers. Note that the LHC and RHC "counters" accumulated sums; in ENIAC we called them "accumulators."

We'll explain the wiring of Figure 15 in terms of a payroll calculation. There is a punched card for each employee, with A being the number of hours, B the hourly rate, and C and D fixed amounts to be added to the paycheck. Point ① shows that the brushes which pick up A and B are wired to the ier and icand counters, respectively. At point ② the factors C and D are transferred from the brushes to RHC and LHC; the position of these wires depends on the locations of the decimal points. In the middle of the plugboard another occurrence of ② shows that a 1/2 entry is wired into one of the partial product counters. This 1/2 produces a rounded digit in the position to the left, the contents of its position and the positions to the right are deleted from the answer. ENIAC used the same method of round-off.

At point ③ the stepwise multiplication results are wired into RHC and LHC. Point ④ shows how some control wires are to be plugged in. I'll explain two of the connections. The multiply-crossfoot switch is wired to accomplish crossfooting (the addition of C and D to AxB) rather than for mere multiplication (AxB alone). Also, RHC and LHC are wired so their contents, the right- and left-hand components, will be combined at the end. Point ⑤ is located near the middle of the diagram. It shows wiring that carries the result $AxB+C+D$ to the brushes and to a summary counter which accumulates $AxB+C+D$ for each employee, and hence registers the total payroll.

The wiring of Figure 15 is deceptively simple, because groups of digit wires are represented by single lines. For example, at upper left the brushes B21-B25 are joined by a single line, an arrow from B23 goes to the multiplier counter digit position E6, and positions E4-E8 are connected by a horizontal line. This symbolism means that brush B21 (reading column 21 of the card) is to be connected to multiplier counter digit position E4, and similarly for the other five digits of the ier. Thus a single arrow represents a bundle of six digit wires. A typical IBM plugboard was somewhat of a mess, as was an ENIAC

problem set-up.

Let us now see how ENIAC was set-up or programmed to solve a specific problem. This was a dual task: digital communication paths were established among the numerical circuits of the units, and a program proper was set up on the program controls. To avoid ambiguity we will call the first part "numerical programming" and the second part "programming proper." Except for the branch operation, these two parts of a program were separate. We will now discuss them in turn.

Each of the ENIAC computing units had input and output sockets for ten digit signed numbers. These inputs and outputs were connected to digit trays by jumper cables. The cables are prominent in Figure 12, where they go from the top of the accumulator program panels to the trays above. They are on the left in Figure 6. The jumper cables and trunk lines are shown schematically at the top of Figures 16 and 17.

Interconnecting the numerical circuits of ENIAC was analogous to setting up the differential analyzer. Both operations established data communication channels between computing units. But there was an important difference. In the differential analyzer, all shafts rotated simultaneously. In ENIAC, the digital channels were used as needed, according to instructions set on local program controls. An accumulator, for example, had five input sockets (labeled α, β, γ, δ, and ε) and two output sockets (one for "add" and one for "subtract"). When setting up an accumulator program control the operator set a switch to one of the eight positions α, β, γ, δ, ε, A, S, AS. The "AS" position caused the accumulator to transmit simultaneously over both its add output and its subtract output, so it could add into one accumulator and subtract into another at the same time.

The main function of a numerical program of an ENIAC was to establish digital communication channels among the units. An auxiliary function was to arrange for some simple numerical transformations. Mechanical "shifters" and "deleters" could be inserted into digital sockets. A shifter was a plug-socket combination in which the wires were shifted to effect a multiplication or division by a power of ten. It was the digital equivalent of the fixed gear of the differential analyzer. To round-off the number in an accumulator the operator did two things. She set a switch to clear the accumulator to "five" in one decade position. And she inserted deleters in the outputs to delete the unwanted digits.

Two adjacent accumulators could be interconnected for double precision

(20 decimal digits).

This concludes our explanation of the numerical part of an ENIAC program. This part determined the precise effect of the instructions set on local program controls, so it is analogous to microprogramming. We turn next to the second part of an ENIAC set-up, the programming proper. This involved interconnecting ENIAC control circuits and setting switches so that these circuits would direct the numerical circuits to perform the desired computation.

Each of the ENIAC computing units had several program controls. We will describe first how the programmer used them and then how they controlled the numerical circuits. The reader should refer to the lower parts of Figures 16 and 17 for schematic representations of ENIAC programs. Figure 17 pictures part of a chart showing how to set up ENIAC to integrate a trajectory by the Heun second-order method. This was a program which I had drawn up in the spring of 1944, while ENIAC was still being designed.

A typical program control had an input socket, some switches on which the instruction was set, and an output socket. When stimulated with a program pulse the program control carried out the instruction set on its switches, and then emitted a program pulse. These program pulses were transmitted around the machine on a system of trays and cables running below the program panels. A subroutine was established by interconnecting local program controls in parallel and in series and setting the switches to specify the operations wanted.

I'll illustrate this process by showing how ENIAC solved the very simple differential equation

$$\frac{dy}{dx} = y \ ,$$

using the first-order difference equations

$$\Delta y_i = y_{i-1}\Delta x$$

$$y_i = y_{i-1} + \Delta y_i .$$

The program calculates 100 values of $y_i (i = 1,2,\ldots,100)$ starting from the initial value y_0. The interval of integration Δx is chosen to be a power of ten (e.g., .01) so the multiplication $y_{i-1}\Delta x$ can be done with a mechanical shifter.

The program has two subroutines, one for inputs (lines ② and ③) and one for calculating and printing (lines ④, ⑤, ⑥, and ⑦). Each subroutine is started by a pulse from the master programmer, and when each is finished it returns a pulse to the master programmer (on line ①). The oper-

acor starts the program by pushing a button on the initiating unit. The pulse produced goes to the master programmer on line ①. From there it goes to line ② to start the input subroutine.

The input subroutine has two sub-steps. The pulse on line ② goes to the card reader program control, which causes the card reader to read a card containing y_0 and Δx. The output pulse on line ③ then goes to instruct the constant transmitter to transmit y_0 and also to instruct accumulator no. 1 (or program control #1) to receive y_0. The output from the latter on line ① goes back to the master programmer.

The master programmer then executes the calculation and print subroutine 100 times, for steps $i = 1,2,\ldots,100$. At the beginning of step i the contents of the accumulators are:

No. 1 holds y_{i-1}

No. 2 holds x_{i-1}

No. 3 is clear.

This subroutine has four sub-steps, which I will identify by their program input lines.

④ y_{i-1} is transmitted from accumulator no. 1 (control #3) and received by accumulator no. 3 (control #1).

⑤ Accumulator no. 3 (control #2) transmits y_{i-1}, which becomes $y_{i-1}\Delta x$ (i.e., Δy_i) after being shifted two positions to the right by a mechanical shifter. Accumulator no. 1 receives Δy_i (control #2) and adds it to y_{i-1}, to make the new value y_i.

⑥ The constant transmitter sends Δx and accumulator no. 2 (control #1) adds it to x_{i-1} to obtain x_i.

⑦ The printer now punches the values of y_i (accumulator 1) and x_i (accumulator 2) on a card.

The output pulse goes on ① to the master programmer, which stops the computation after the step $i = 100$ is completed.

The role of the master programmer in an ENIAC program should be clear from this example. The master programmer had input and output sockets for program pulses, six-stage counters for controlling subroutines in sequence, and banks of decade counters for counting the number of times a subroutine had been executed. The desired numbers were set on switches. By these means it could operate a set of subroutines in sequence, causing each to be executed a fixed number of times (as preset on switches) or until a branch occurred.

Overall, the master programmer had electronic facilities for managing a large and rather complicated structure of subroutines.

For the branch operation the sign digit of a number was converted into a program signal by an otherwise unused (dummy) program control and then fed into a program channel. A negative sign resulted in a program pulse, which caused the master programmer to shift to another subroutine. A positive sign produced no program pulse, so on this alternative the master programmer continued to execute the same subroutine.

This completes our description of ENIAC programming proper. We'll next make a few remarks on the control circuits of a typical ENIAC unit. These circuits were of two kinds: the local program controls, and the common control circuits. Through its program switches, each local control of a unit operated the common control circuits of the unit, and the common circuits in turn operated the numerical circuits.

A simplified accumulator program control circuit is shown in Figure 5. A pulse on its input set the flip-flop, which remained on during the execution of the instruction, in this case for one addition time. Tubes 3 and 4 are the flip-flop proper, and tubes 1 and 2 are used to set and reset it. When it is set the flip-flop performs the following switching actions. Through tubes 6 and 7 it activates the common control circuits for receiving a number, transmitting a number (for adding), or transmitting the complement of a number (for subtracting), according to the switch setting. Through tube 8 it operates common circuits for clearing or not clearing the accumulator after transmission, according to the switch setting. The flip-flop also turns on gate 9, which then passes a program pulse from the cycling unit at the end of the addition period. This pulse resets the flip-flop. It may also go through tubes 10 and 11 to a program line, where it will initiate the next step of the program.

Let us summarize our discussion of ENIAC programming. It had two parts: numerical programming and programming proper. The method for each was mechanical: plugging in cables and wires and setting switches by hand. This method was derived from the IBM plugboard.

Numerical programming was analogous to and derived from the process of setting up the differential analyzer to solve a problem. Programming proper involved interconnecting ENIAC control circuits and setting switches so that these circuits would direct the numerical circuits to perform the desired computation. Subroutines were established on the local program controls of the computing, input, and output units of ENIAC. The master programmer was then

set up to orchestrate these subroutines into a single master routine for solving the problem.

5. Evaluation of ENIAC

We have covered the antecedents and nature of ENIAC, its hardware, arithmetic design, organization, and method of programming. After summarizing the results I'll take stock and see what ENIAC added to the history of computers. I'll conclude that its main contribution was a hardware revolution: the successful employment of vacuum tubes to make a fast, reliable, powerful, general-purpose, scientific, digital computer.

We saw in Section 3 that ENIAC's organization derived from the differential analyzer. To a person familiar with modern computers, this organization does not seem natural or simple. Let me illustrate this point with a story.

Once I gave a lecture on ENIAC and wanted to display a program tray, a heavy metal object eight feet long. A computer science student was nearby and I asked him to help me carry it to the lecture hall. He looked very puzzled and uninterested, and I could see him thinking: That piece of junk! What's that for? So I said: That's a communication channel of ENIAC. He continued to stare at me, so I added: ENIAC was the first general-purpose electronic computer. He remained unimpressed. Finally I said: That's part of the ENIAC, the first *multiprocessor*. His face lighted up, and he rushed to help me!

However, ENIAC was not really a multiprocessor in the modern sense. No ENIAC unit could compute by itself, just as no unit of the differential analyzer could compute by itself. A single full electronic arithmetic unit did not come until the first stored program computers.

The original aim of the ENIAC project was to build a digital electronic version of the differential analyzer. Later, it was realized that this would be a general-purpose scientific computer. The following question was not asked at the time, but it is relevant here. Various electronic computing organs were available: counters, flip-flops, switches, and resistor matrices. How could these organs best be organized to make a large, powerful, general-purpose computer? ENIAC's organization was not a simple and efficient answer to this question. A logical organization more like that of the first stored program computers would have produced a smaller and better machine. But even with ideal designs ENIAC would have been a very large machine, perhaps 50 times as large as any known electronic system.

We saw in Section 4 that ENIAC was programmed by a mechanical technique

derived from the IBM plugboard. For comparable problems, ENIAC was easier to program than the differential analyzer, because wires and switches were easier to manipulate than shafts and gears. However, an electromechanical method of programming would have been superior. For example, programs could have been set up on a central bank of stepping switches and relays. For a specific problem these could have been positioned by electrical signals from a paper tape or deck of printed cards.

In their 1943 ENIAC proposal, Eckert and Mauchly mentioned the possibility of centralized control by punched cards. Referring to their Figure 1 (our fig. 11) they said:

> For completeness, the diagram includes a "Program Selector" to emphasize that centralized control by punched cards or other means may be used if desired. Such a selector may, for instance, be used to vary initial conditions from one run to the next, or *even* to change the sequence of operations and thereby the equation being solved. [Italics added.]

This was not achieved in ENIAC.

In evaluating ENIAC, one should keep in mind that it was developed during World War II. The immediate goal was to calculate firing tables. The long range goal was a general-purpose computer. ENIAC's organization and method of programming served this purpose without being novel. Moreover, because of the emergency character of the war, ENIAC was developed very rapidly. The proposal was written in April, 1943. Work on electronic counters began in June. The prototype accumulators were working and the basic logical design of the whole machine completed by the summer of 1944. ENIAC began to solve its first problem in December, 1945. This was an important problem from Los Alamos, still classified. By February, 1946 ENIAC completed this problem, and had been demonstrated publicly. The whole period was less than three years.

The engineering of ENIAC was most excellent. This was due primarily to Pres Eckert. He led in the design of fast circuits and saw the importance of reliability for such a large and novel system.

Rigid safety factors were imposed on all electronic designs; for example, a factor of three on voltage swings, a factor of two on switching times. The whole system was constructed from a few basic circuit types. Circuits were mounted on removable plug-in units whenever the number of inputs and outputs were sufficiently small. Components were carefully selected, pretested, and operated below their standard ratings. We always gated short pulses (about 3 microseconds) against long pulses (which we called gates), never short pulses against short pulses.

Several diagnostic modes of operation were incorporated in the system. Just as the independent-variable motor of the differential analyzer could be slowed down to improve the accuracy of the computation, so the cycling unit of ENIAC could be slowed down to insure correct operation or for diagnostic purposes. In addition, the operator could set the cycling unit to operate in a control mode. When it was in this mode she could call for clock pulses as she wanted them, either in cycles of one addition time, or pulse by pulse. All ENIAC circuits were designed to hold their information in static form whenever the cycling unit stopped. Also, all flip-flops and counters had neon lights attached to each stage, so their states were easily ascertained.

The master programmer could be rearranged so that the program was divided into small segments operable independently. By all these means, the operator could step the computation along, program segment by program segment, addition by addition, pulse by pulse, until she located the error. Also, the parallel programming facilities of ENIAC allowed her to have test routines on the machine to be run periodically.

These design precautions were sufficient. ENIAC needed to be carefully maintained and operated with suitable precautions. It turned out, for example, that whenever the heaters of the vacuum tubes were turned on several tubes would fail, so it was necessary to always keep the heaters on. After it was completely debugged and when it was properly maintained, ENIAC operated with good reliability.

The 18,000 vacuum tube ENIAC proved that electronic technology could be used to make fast, powerful, reliable, general-purpose computers. Its speed of computation was three orders of magnitude greater than the speed obtainable with electromechanical technology, and it solved problems hitherto beyond the reach of man. The electronic design of ENIAC was optimal, and led naturally to the stored program computer.

ENIAC constituted a hardware revolution in computers. It took them from the electromechanical to the electronic. It began our modern age of electronic computers.

6. High-Speed Read-Write Electronic Stores

ENIAC had fast electronic circuits for storage, arithmetic, and control, and executed complicated programs rapidly. However, its storage capacity was small and its programming procedures mechanical. The technology was available

for programming it electromechanically, but even then its programming equipment would have been one full level of technology below its electronic computing circuits.

Both the storage and programming deficiencies of ENIAC were eliminated by its successors, the first generation of stored program computers. These machines put programming on a par with arithmetic and control, and thereby began a second revolution in computing.

I shall divide the development of the stored program computer into two historical stages, the preliminary design stage (1944-46) and the final design and construction stage (1946-52). The preliminary design stage had two related parts, the development of hardware for storing information, and the design of an order code and suitable organization for the stored program computer. The final design and construction stage saw the development and completion of many stored program computers, those of the EDVAC family (EDVAC, EDSAC, UNIVAC I, SEAC, SWAC, etc.) and those of the IAS family (IAS, Whirlwind, ILLIAC, JOHNNIAC, IBM 701, etc.). I will not have time to discuss this stage.

From the beginning, designers of electronic computers were aware of the need for a storage device superior to the vacuum tube register or counter. Atanasoff invented his rotating drum with capacitors as a solution to this problem. Both Crawford and Rajchman invented the resistor matrix function table store, but this was only a partial solution, since it was a read-only memory. Perry Crawford made an important contribution when he proposed a cyclic magnetic disk memory. Pres Eckert realized that a computer based on such a memory would be superior to ENIAC. Early in 1944 he proposed a numerical calculating machine based on rotating magnetic discs or drums.

The next step in the development of electronic memories occurred when Pres and John combined the cyclic idea of drum and disc memories with the supersonic delay line being used to time radar signals. To measure the elapsed time of a radar pulse from the antenna to the airplane and back to the antenna, another pulse was sent down a delay line and reflected back. Pres had earlier worked on these lines at the Moore School for the MIT Radiation Laboratory.

Figure 18 shows the delay lines of Maurice Wilkes' Cambridge EDSAC, the first stored program computer to operate. I'll explain how the delay line memory worked, using the schematic diagram of Figure 19.

Electronic circuits were connected in a cycle with a tube of mercury which had a quartz crystal at each end. A quartz crystal is a piezo-electric device which will convert electrical pulses into physical or acoustic vibrations, and

vice versa. An electrical pulse from the circuits caused the input crystal to vibrate, producing an acoustic pulse in the mercury. This acoustic pulse traveled down the tube and caused the output crystal to vibrate, so it produced an electrical pulse. The electrical pulse returned to the electronic circuits, where it was retimed by reference to a clock pulse and reshaped. The absence of a pulse would cycle similarly.

Preliminary design work was done by Pres and Kite Sharpless. Measurements indicated that 1000 pulse positions (bits), spaced one microsecond apart, could be stored in a single delay line. About ten vacuum tubes (envelopes) were needed to reshape and retime pulses and to switch pulse streams in and out of memory, so that 1000 bits could be stored at the cost of about 10 envelopes. Moreover, because of the serial nature of the delay memory, fewer switching tubes were required for entering and recovering information than in flip-flop or counter memories. Thus the new memory was better by more than a factor of 100 to 1! It was now possible to store program information, as well as data, electronically and at high speed.

In the late summer of 1944, not long after Pres and John invented the mercury delay line store, von Neumann conceived of using an electron beam oscilloscope for computer storage. Information is placed on the surface of an iconoscope tube by means of light coming from the outside; this information is then sensed by an electron beam. Johnny thought that information could also be placed on the inside surface of such a tube by having an electron beam either deposit a charge in a small area or not. The recorded information would later be read by means of the same beam. Charge would leak from one area to the next, so the information would need to be periodically refreshed, as in Atanasoff's drum memory.

This idea was not worked out at the time. It turned out to be more difficult to reduce to practice than the mercury delay line, but electrostatic stores were later developed by Jan Rajchman and, independently, by F. C. Williams in England.

Thus two forms of high-speed, read-write memory of sizeable capacity were conceived in 1944: the cyclic mercury delay line store and the random-access electrostatic store.

7. Code and Organization for the Stored-Program Computer

At the Moore School we then planned to build a stored program computer, the "EDVAC", for "Electronic Discrete Variable Arithmetic Computer." It would

have a large mercury delay line memory, at least 1024 words of 32 bits. In comparison, ENIAC had 20 words of variable storage, about 400 words of read-only store, and the equivalent of perhaps 200 instructions, mechanically set.

Pres and John had devised ways of operating circuits at a megacycle pulse rate, matching pulse against pulse for switching at this rate. Serial addition could be done in 32 microseconds, as fast as the numbers circulated. Multiplication by repeated addition would be several times faster than "high-speed" ENIAC multiplication, and would match well with the waiting time for the circulating delay memory.

Because EDVAC would be so much faster, smaller, and simpler than ENIAC, there was no longer the need for parallelism to gain speed, and it was decided to store numbers serially and process them serially. The guiding principle of EDVAC design was: one thing at a time, down to the last bit!

The delay-line store was to be used for both numbers and orders. In his earlier proposal for a numerical calculating machine based on magnetic discs or drums, Pres had proposed storing program information on some discs or drums. Since the storage of instructions and data in the same device is unique to the stored-program computer, it is important to be clear on our concept of a "program" at that time.

The Harvard Mark I and the Bell Laboratories' machines used electromagnetic components for storage and computation. They, like ENIAC, were limited in their read-write storage capacity. Orders for these machines were expressed in binary coded form and punched into paper tape. Paper tape is a read-only memory, and the orders punched into it referred to storage bins by means of fixed addresses. Thus these machines used fixed or constant address order codes.

In March of 1945 von Neumann spent two days at the Moore School having extended conversations with Pres, John, Herman, and me about EDVAC. These meetings were devoted mostly to the hardware design and local organization of the memory and the arithmetic equipment. EDVAC was to have a memory unit of mercury tanks, arithmetic equipment, magnetic tapes for input and output, and control equipment. There were to be one or two switches to transfer instructions and numbers from memory tanks to the computing equipment. There were to be serial adding, multiplying, and dividing circuits, fed by short delay lines.

After these meetings Johnny went off and wrote a draft report on the design of EDVAC. Without his knowledge, this was issued as *First Draft of*

a Report on the EDVAC, Moore School of Electrical Engineering, June 30, 1945. Undoubtedly he would have given credits to others. In my personal opinion these would have gone primarily to Pres Eckert and John Mauchly, and secondarily to Herman Goldstine and myself.

I'll give a brief description of von Neumann's EDVAC organization and order code. For this description I'll draw on my Introduction to his *Theory of Self-Reproducing Automata.*

The basic internal units of EDVAC were a high-speed memory M, a central arithmetic unit CA, and a central control CC. For communication there was an outside recording medium R, and input organ I, and an output organ O.

The memory M was to be composed of possibly as many as 256 delay lines each capable of storing 32 words of 32 bits each, together with the switching equipment for connecting a position of M to the rest of the machine. The memory was to store initial conditions and boundary conditions for partial differential equations, arbitrary numerical functions, partial results obtained during a computation, etc., as well as the program (sequence of orders) directing the computation.

The outside recording medium R could be composed of punched cards, paper tape, magnetic wire or tape, or photographic film, or combinations thereof. It was to be used for input and output, as well as for auxiliary low-speed storage. The input organ I transferred information from R to M; the output organ O transferred information from M to R. The notation of M was binary; that of R was decimal.

The central arithmetic unit CA was to contain some auxiliary registers (one-word delay lines) for holding numbers. Under the direction of the central control CC it was to add, subtract, multiply, divide, compute square-roots, perform binary-decimal and decimal-binary conversions, transfer numbers among its registers and between its registers and M, and choose one of two numbers according to the sign of a third number. The last operation was to be used for transfer of control (jumping conditionally) from one order in the program to another.

The first bit of each word was zero for a number, one for an order. There was a single switch connecting the memory to the rest of the machine. This switch sensed the first bit of a word coming from memory and on that basis routed numbers to the central arithmetic unit CA and orders to the central control CC. Eight bits of an order were allotted to the specification of the operation to be performed and, if a reference to M was required, thirteen bits

to an address.

Normally orders were taken from the delay lines in sequence, but one order with address z provided for the central control CC to take its next order from memory position z. This was the unconditional shift of control. The conditional shift of control was based on the central arithmetic unit CA's ability to choose one of two numbers according to the sign of a third number. The first two numbers were the addresses of the orders which were to be executed according to whether the condition on the third number was or was not satisfied.

Numbers were processed in CA serially, the least significant bits being treated first, and only one operation was performed at a time. when a number was transferred from CA to address w of M, account was taken of the contents of w; if w contained an order (i.e., a word whose first bit was one), then the 13 most significant bits of the result in CA were substituted for the 13 address bits located in w. The addresses of orders could be modified automatically by the machine in this way, leaving the operand part of the instruction untouched.

This last feature of Johnny's order code was crucial, and was invented by him to facilitate programming in the new machines. These new machines would be capable of storing large quantities of data. Von Neumann saw that while the fixed address order codes of the Harvard and Bell machines were adequate for the limited memories of these machines, they would not be efficient for the large memories of the new electronic machines.

Von Neumann's solution to this problem was to invent the modern variable address code. In his EDVAC code variable or virtual addresses are stored in orders. The program is then written so that the machine calculates and substitutes a specific address into an order before each specific execution of that order. This substitution process is controlled by branching when the variable address reaches a preset bound. Thus von Neumann designed his EDVAC code to make possible the recursive loops we are all familiar with. What today we call indexing and relative addressing was accomplished in EDVAC by processing addresses in the central arithmetic unit CA.

It is worth comparing the way EDVAC and ENIAC handled different uses of arithmetic in computing. In ENIAC, the primary arithmetic of solving a problem was done in accumulators, working with the multiplier and the divider square-rooter. Function table look-up was accomplished by local arithmetic and switching circuits in the function table units. The master programmer of ENIAC also did arithmetic. It used counters to count the number of times a subroutine was used, and caused a branch when a limit number (preset on switches) was

matched. In EDVAC, all these arithmetic functions were centralized in the one central arithmetic unit CA.

In summary, von Neumann was the first to see and exploit the fact that when orders or instructions are stored in a high-speed read-write electronic memory, they can be manipulated arithmetically and modified by the machine itself. His variable address EDVAC code was the basis of the modern computer software revolution.

I'll make one last point about Johnny's logical design of EDVAC. This concerns the separation of logic from electronics. In ENIAC these were mixed. A gate tube performed the logical operations of "not-and" ("nan"), but it could only drive so much capacitance in the allowed time before an amplifier was needed, and the amplifier was a logical "not". The logical design of ENIAC proceeded pari passu with the electronic design. As a consequence, the logical design of ENIAC was not really completed until the circuit design was finished.

In contrast, von Neumann worked out most of the abstract logical design of EDVAC in his draft report. He did this by using idealized switches with delays, derived from the logical neurons of Warren McCulloch and Walter Pitts. This abstraction of logic from engineering enabled him to do the logic of EDVAC without simultaneously doing the engineering, and thereby made it possible for him to essentially complete the design in one draft. This also simplifies the historian's task, for it makes sharper the division between the preliminary design stage (1944-46) and the final design and construction stage of the stored-program computer (1946-52).

We all knew that von Neumann's logical designs were realizable because he had worked out the building blocks with the group before he wrote the report. I remember well a meeting in the spring of 1945 at which we discussed serial adders. Pres and John had designed several serial adders, the simplest of which took ten tubes. Not knowing of these results, von Neumann announced cheerily that he could build an adder with five tubes. We all looked amazed, and Pres said, "No, it takes at least ten tubes." Johnny said, "I'll prove it to you," rushed to the board, and drew his adder.

"No," we said, "your first tube can't drive its load in one microsecond, so an inverter is needed, then another tube to restore the polarity." And so the argument went. Johnny was finally convinced. But he was not taken aback. "You are right," he said. "It takes ten tubes to add — five tubes for logic, and five tubes for electronics!"

In his *First Draft of a Report on the EDVAC* von Neumann also proposed the development of a high-speed memory using an electron beam oscilloscope. He thought such a store would be superior to the delay line store because all the storage cells in a cathode ray tube would be directly accessible.

Late in 1945 von Neumann decided to build a machine at the Institute of Advanced Study based on an electrostatic storage tube to be developed at RCA Laboratories by Jan Rajchman. The rest of the computer was to be designed and built at the Institute. Jan's storage tube was called the "Selectron". In the end the Williams tube electrostatic memory was developed before the Selectron, so the IAS machine used it. One machine of the IAS family, the JOHNNIAC at RAND Corporation, used Selectrons. For the story of this and many other topics I have discussed see Herman Goldstine's *The Computer from Pascal to von Neumann* (Princeton University Press).

In the spring of 1946 Johnny, Herman and I worked out the logical design of the Institute for Advanced Study computer (IAS). Our report, *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*, was issued on June 28, 1946.

To take advantage of the gain in memory speed of Selectrons over delay lines, we planned to store the bits of a word in parallel, one bit per Selectron, and process them in parallel. This required more arithmetic equipment than a serial arithmetic unit, but it saved the control equipment needed for timing serial operations. With 40 bit words, two orders could be placed in one word, an important economy. It seemed that in these ways the IAS machine would be an order of magnitude faster than EDVAC and no more complex.

Figure 20 lists the 21 internal orders of the IAS machine. "S(x)" denotes the word stored or to be stored in Selectron address x. There were to be 4096 words, stored in 40 Selectrons. There was a register associated with this memory, called the "Selectron register". Words were moved from the Selectrons to it, and thence to either the arithmetic unit (for a number) or the control (for a pair of orders). Words were moved from the arithmetic unit to the Selectron register and then into the Selectrons. The arithmetic unit contained a 40 bit accumulator and also a 40 bit register.

The leftmost 12 bits of an order were for an address and the rightmost 8 bits specified an operation. There were two orders in a word, left-hand and right-hand. Correspondingly, address substitutions and shifts of control were of two forms, left-hand and right-hand.

The accumulator could add, subtract, or take the absolute value of $S(x)$. It could be cleared or not before these operations. It could halve or double its contents by appropriate shifts. Multiplication and division involved both the accumulator and the arithmetic register. The contents $S(x)$ could be transferred to the arithmetic register, and the arithmetic register would transfer to the accumulator. The accumulator could transfer to $S(x)$, either totally (for numbers) or partially (for substituting addresses in orders). We decided not to have orders for square-rooting and conversions between binary and decimal, but to program these operations.

This completes my description of the preliminary design of the IAS computer. Its order code and logical design was, I think, superior to that of the machines based on the serial mercury delay line memory.

8. Conclusion

I will conclude by summarizing briefly the state of the stored program computer at the end of the preliminary design stage in the summer of 1946. The mercury delay line memory was known to be workable, though not yet designed or built. Von Neumann's logical design of EDVAC was available; the electronic design had yet to be done. The Selectron was being developed. The preliminary logical design of the IAS machine had been done; the electronic design remained.

The final design and construction stage of the development of the stored program computer was beginning. Several groups were working at this time, and others started later. Much development and construction work remained, mostly electronic, some logical. The history of this stage (1946-52) would take another long paper.

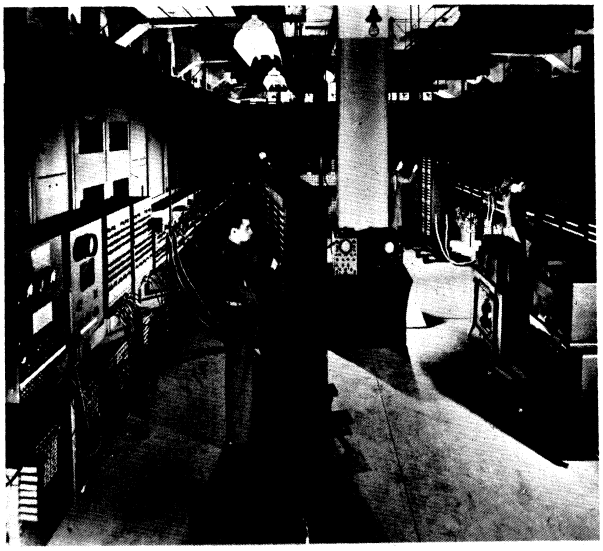
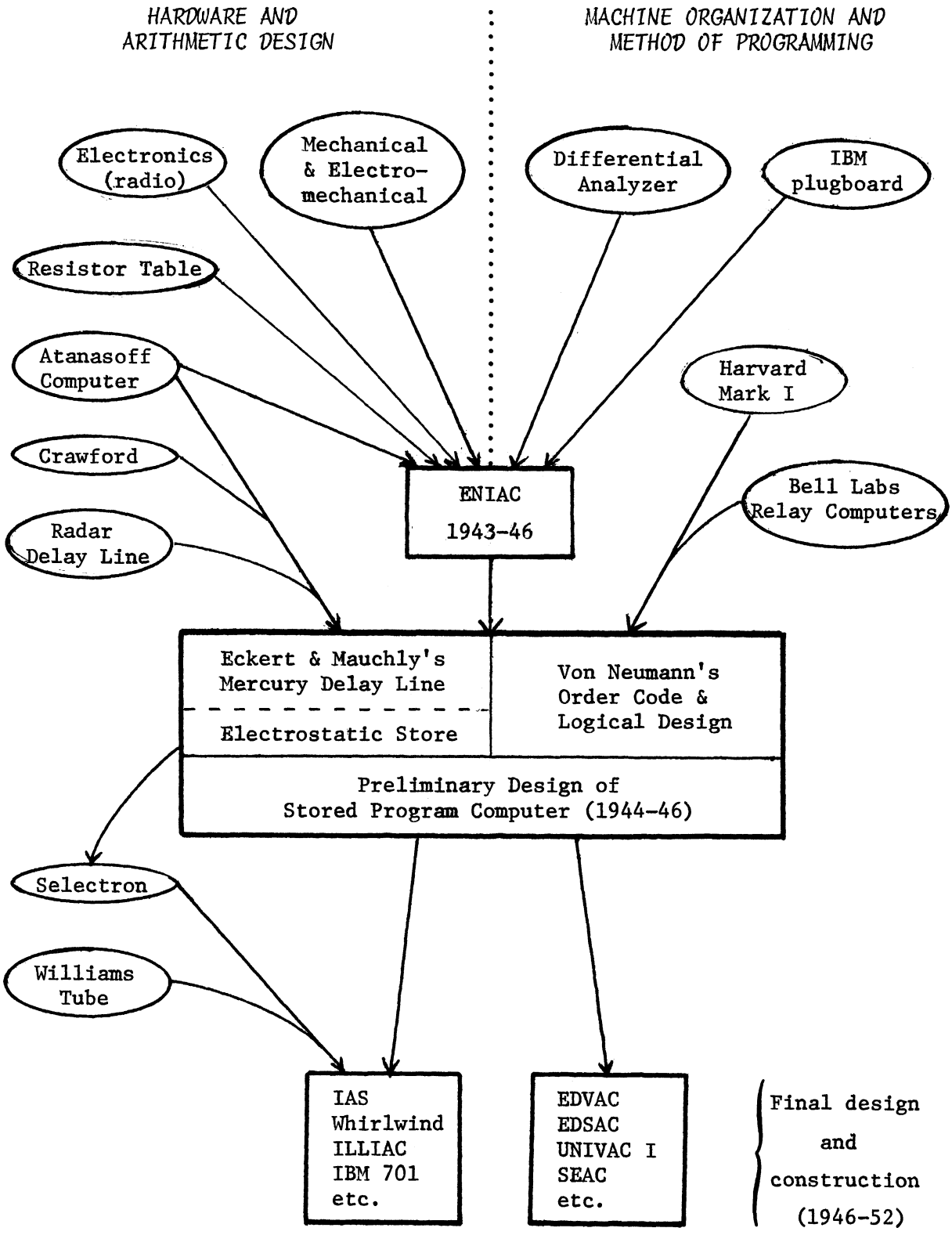
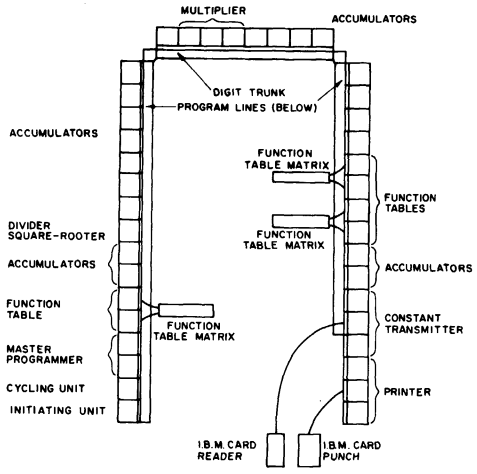


Fig. 1 ENIAC, Overall View

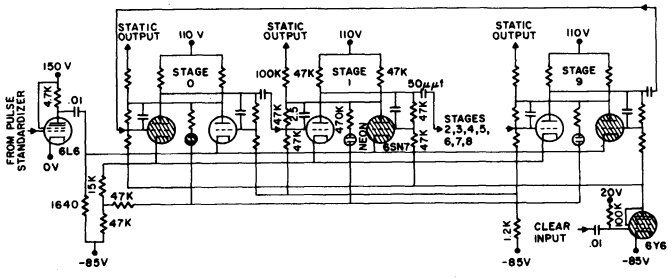
FIG. 2 ORIGIN OF ENIAC AND THE STORED PROGRAM COMPUTER





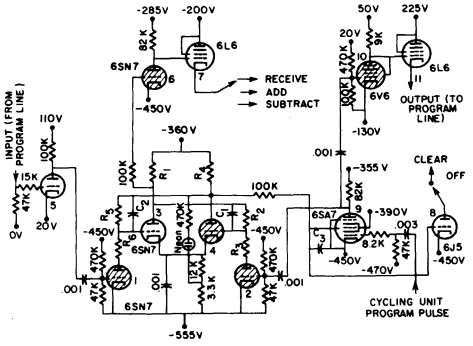
ENIAC LAYOUT

Fig. 3 ENIAC Layout



DECADE RING COUNTER

Fig. 4 ENIAC Decade Ring Counter



ACCUMULATOR PROGRAM CONTROL CIRCUIT
(SIMPLIFIED)

Fig. 5 Simplified Accumulator Program Control Circuit

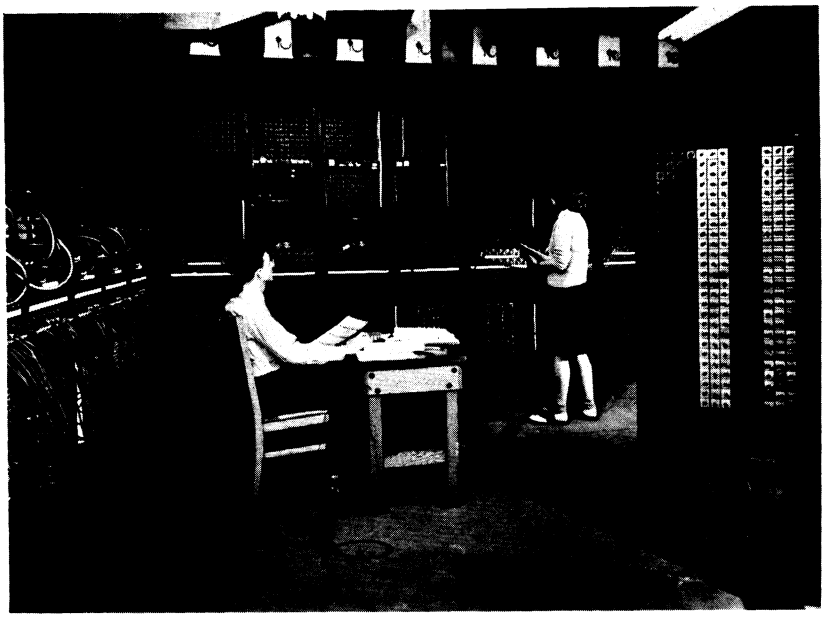
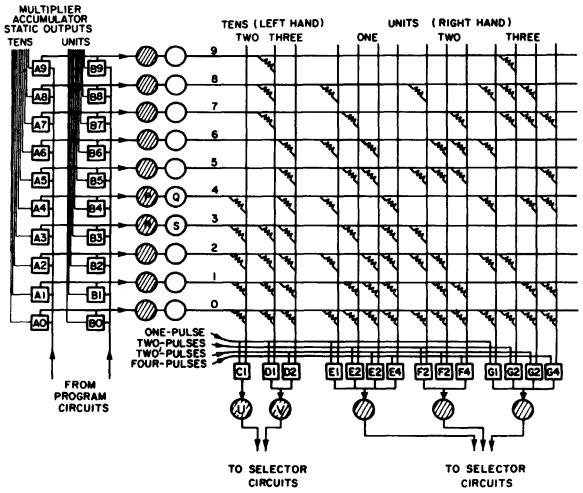
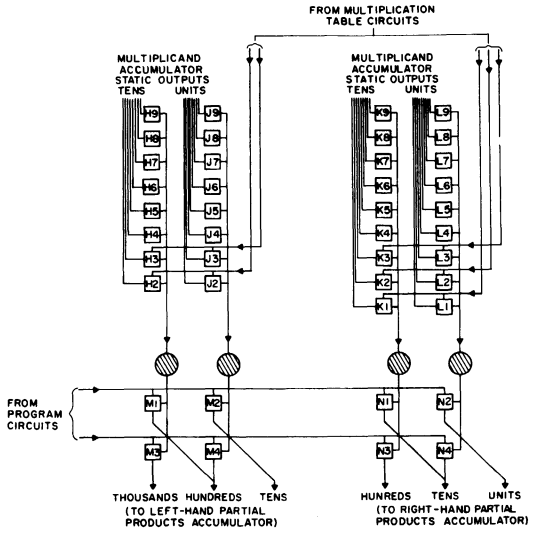


Fig. 6 ENIAC High-Speed Multiplier and Associated Accumulato

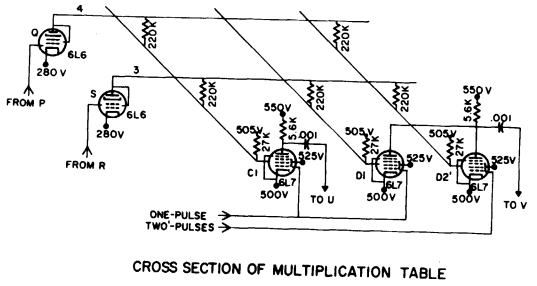


MULTIPLICATION TABLE CIRCUITS



SELECTOR AND SHIFTER CIRCUITS

Fig. 7 Simplified Block Diagram of ENIAC Multiplication Circuits



CROSS SECTION OF MULTIPLICATION TABLE

Fig. 8 Cross-section of ENIAC Multiplication Table

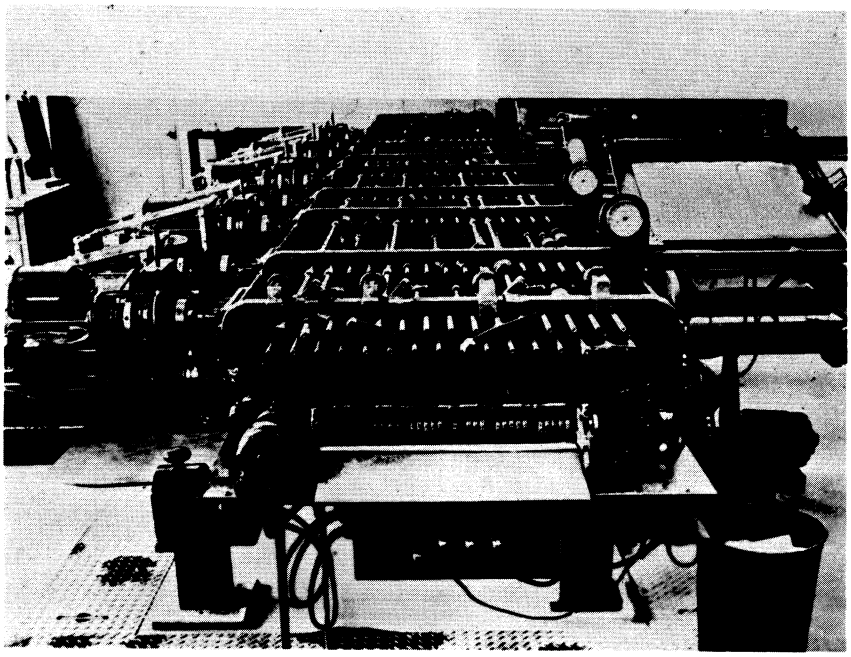


Fig. 9 Aberdeen Differential Analyzer

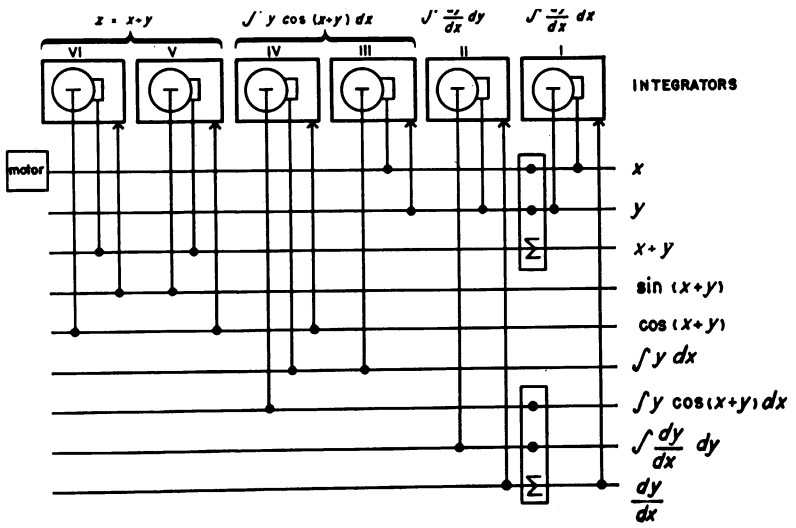


Fig. 10 Set-up Diagram for a Differential Analyzer

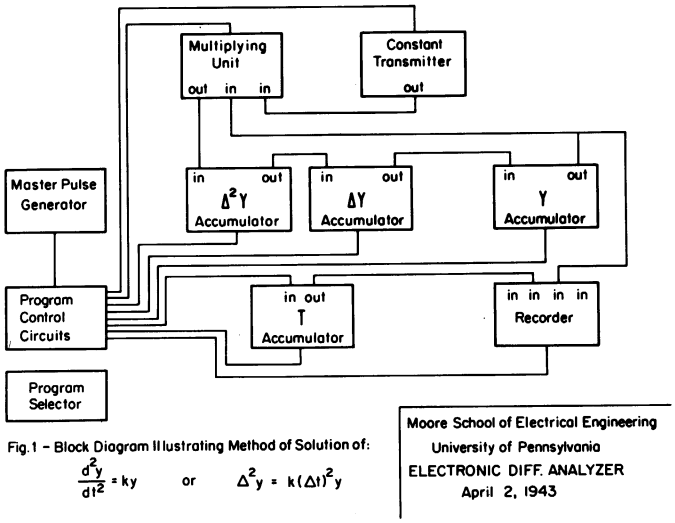


Fig. 11 Figure 1 of Eckert and Mauchly's "Report on an Electronic Diff.* (sic) Analyzer," April 2, 1943

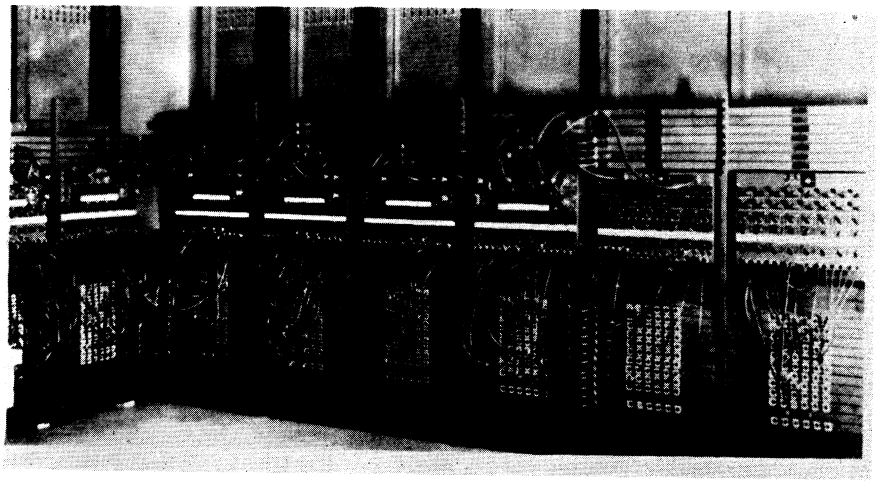
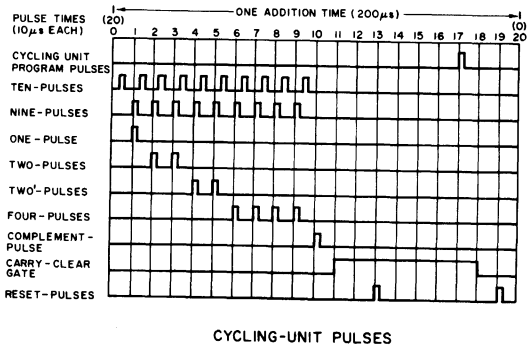


Fig. 12 ENIAC Programming Panels and Cables



CYCLING-UNIT PULSES

Fig. 13 ENIAC Cycling-unit Pulses

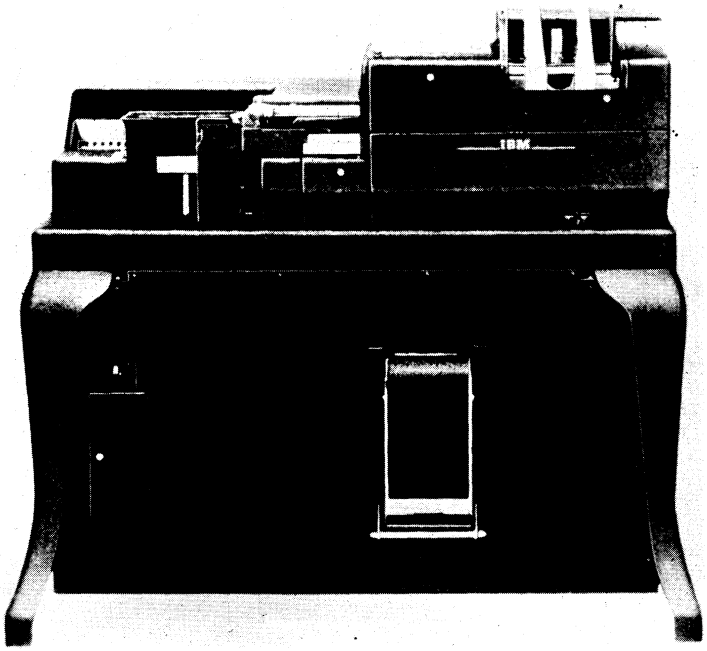


Fig. 14 IBM 601 Crossfooting Multiplier

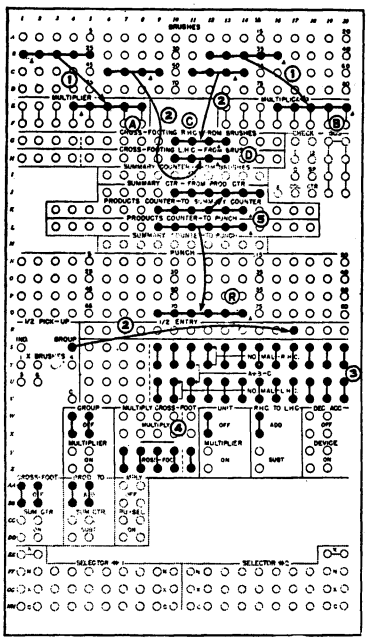
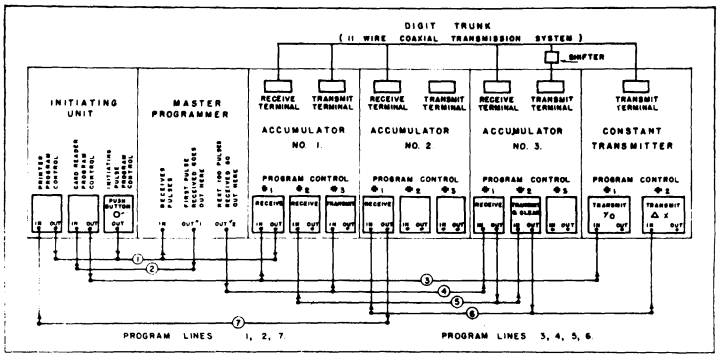


Fig. 15

Wiring Diagram for
IBM 601 Plugboard



Interconnection of units required to solve the differential equation $\frac{dy}{dx} = y$. This uses only a very small part of the ENIAC

Fig. 16 Simplified ENIAC Program-Diagram

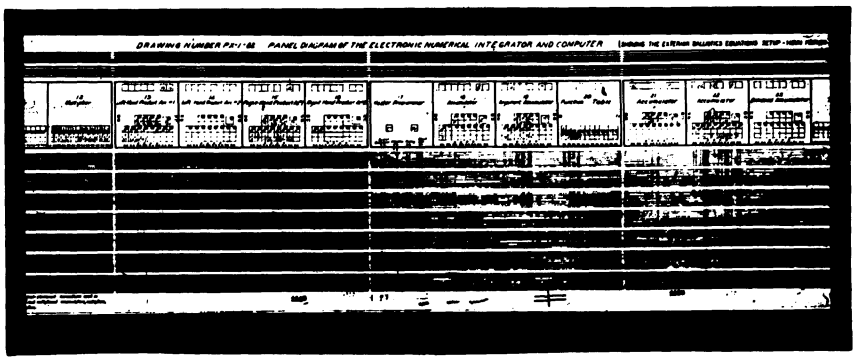
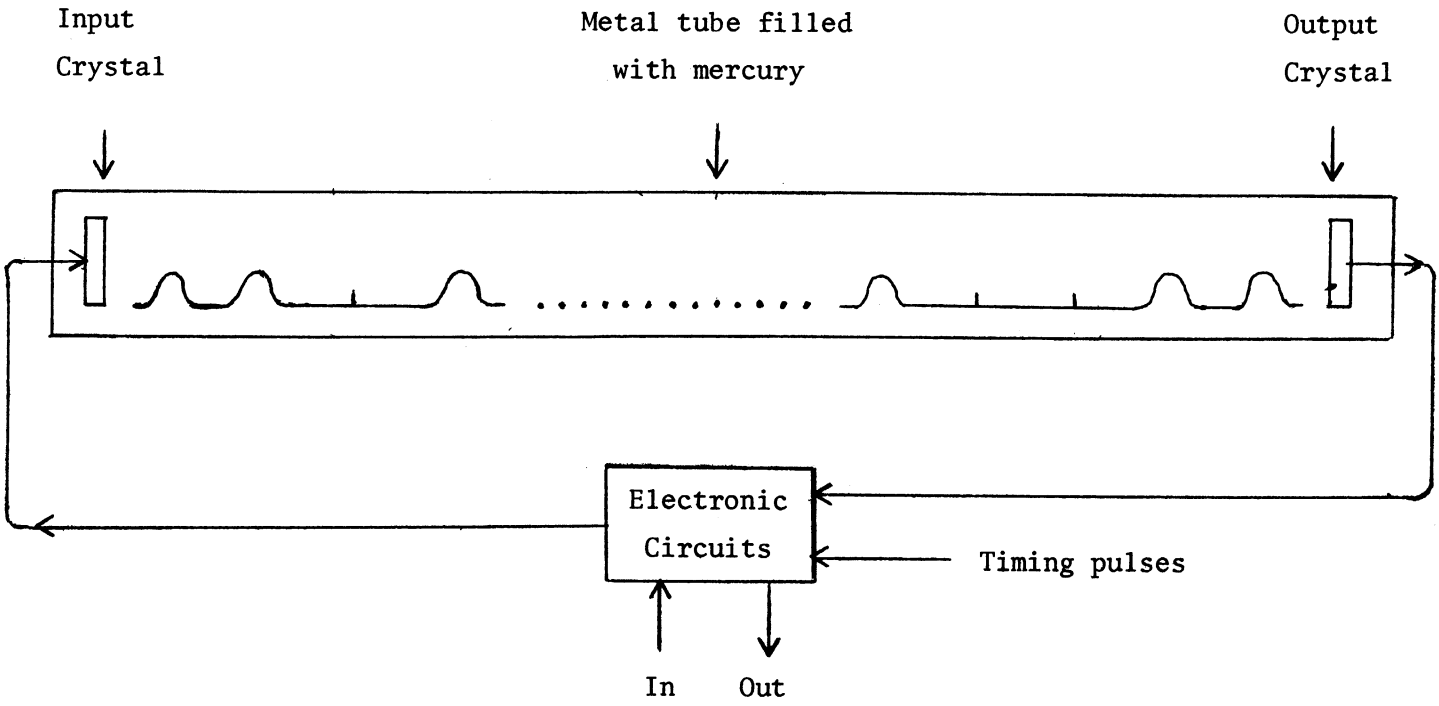


Fig. 17 ENIAC Program for Heun Method Trajectory

Fig. 18 EDSAC Delay Lines



The sequence of acoustic pulses and blanks traveling down the mercury from left to right is depicted schematically.

Fig. 19 Diagram of Acoustic Delay Line Memory

	Symbolization		Operation
	Complete	Abbreviated	
1	$S(x) \rightarrow Ac+$	x	Clear accumulator and add number located at position x in the Selectrons into it
2	$S(x) \rightarrow Ac-$	$x-$	Clear accumulator and subtract number located at position x in the Selectrons into it
3	$S(x) \rightarrow AcM$	xM	Clear accumulator and add absolute value of number located at position x in the Selectrons into it
4	$S(x) \rightarrow Ac-M$	$x-$M	Clear accumulator and subtract absolute value of number located at position x in the Selectrons into it
5	$S(x) \rightarrow Ah+$	xh	Add number located at position x in the Selectrons into the accumulator
6	$S(x) \rightarrow Ah-$	$xh-$	Subtract number located at position x in the Selectrons into the accumulator
7	$S(x) \rightarrow AhM$	xhM	Add absolute value of number located at position x in the Selectrons into the Accumulator
8	$S(x) \rightarrow Ah-M$	$x-h$M	Subtract absolute value of number located at position x in the Selectrons into the Accumulator
9	$S(x) \rightarrow R$	xR	Clear register[a] and add number located at position x in the Selectrons into it
10	$R \rightarrow A$	A	Clear accumulator and shift number held in register into it
11	$S(x) \times R \rightarrow A$	xX	Clear accumulator and multiply the number located at position x in the Selectrons by the number in the register, placing the left-hand 39 digits of the answer in the accumulator and the right-hand 39 digits of the answer in the register
12	$A \div S(x) \rightarrow R$	$x \div$	Clear register and divide the number in the accumulator by the number located in position x of the Selectrons, leaving the remainder in the accumulator and placing the quotient in the register
13	$Cu \rightarrow S(x)$	xC	Shift the control to the left-hand order of the order pair located at position x in the Selectrons
14	$Cu' \rightarrow S(x)$	xC′	Shift the control to the right-hand order of the order pair located at position x in the Selectrons
15	$Cc \rightarrow S(x)$	xCc	If the number in the accumulator is $\geqq 0$, shift the control as in $Cu \rightarrow S(x)$
16	$Cc' \rightarrow S(x)$	xCc'	If the number in the accumulator is $\geqq 0$, shift the control as in $Cu' \rightarrow S(x)$
17	$At \rightarrow S(x)$	xS	Transfer the number in the accumulator to position x in the Selectrons
18	$Ap \rightarrow S(x)$	xSp	Replace the left-hand 12 digits of the left-hand order located at position x in the Selectrons by the left-hand 12 digits in the accumulator
19	$Ap' \rightarrow S(x)$	xSp'	Replace the left-hand 12 digits of the right-hand order located at position x in the Selectrons by the left-hand 12 digits in the accumulator
20	L	L	Multiply the number in the accumulator by 2, leaving it there
21	R	R	Divide the number in the accumulator by 2, leaving it there

[a] Register means arithmetic register.

Fig. 20 Instruction Set for the Institute of Advanced Study Computer