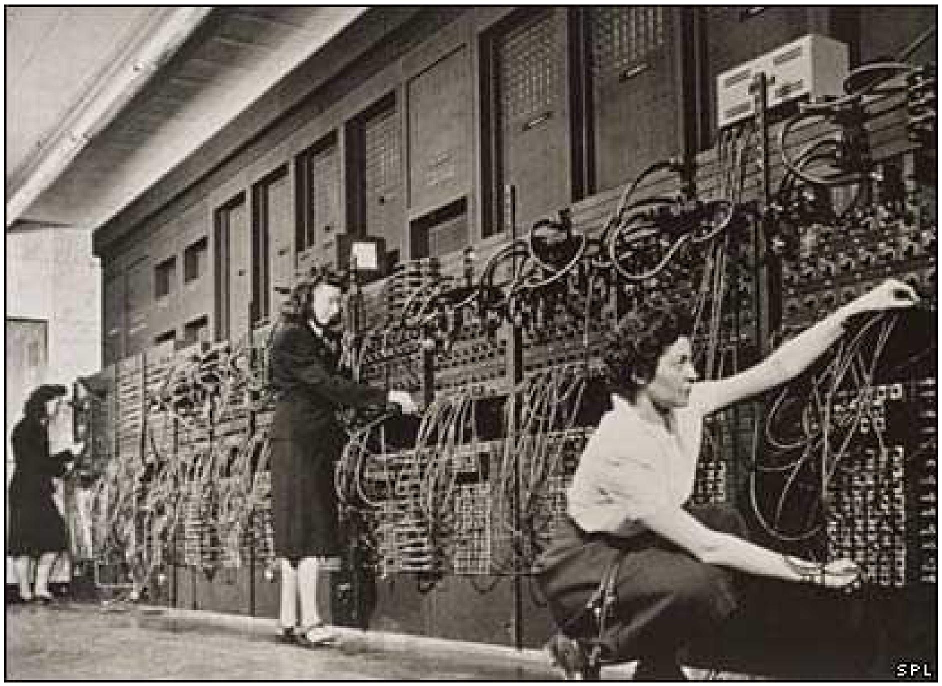
Sensing computers. Developing mathematics from man-computer collaborations in the early years of computing.



Liesbeth De Mol
Centre for Logic and Philosophy of Science, Belgium
elizabeth.demol@ugent.be

Intro.

Introduction

⇒ **(local) goal** Understanding changing human-computer interactions and their impact on math (study of the history of math in relation to history of hard-and software)

⇒ **Motivation**

- Significance and impact of the computer on our society + "hidden" user-adapted computers

⇒ The Heideggerian assumption: "[L]arge sections of computer science are paralyzed by accepting this moron as their typical customer. [U]ser friendliness is, among other things the cause of a frantic effort to hide the fact that eo ipso computers are mathematical machines (Dijkstra, 1985)

⇒ Strategy: "we cannot fully understand our own conceptual scheme without plumbing its historical roots, but in order to appreciate those roots, we may well have to filter them back through our own ideas." (J. Webb, 1981)

⇒ Going back to the roots of digital computing – developing math from "physical" interactions/encounters between ENIAC and mathematicians/logicians

⇒ Confronting the more "physical" collaborations with the now.

Introduction (2): Overview

- “Quick” tour through ENIAC
- Lehmer, number-theory and the explorative attitude in math
- “Johny”, Monte Carlo, the bomb and the flow diagram
- Curry and the composition of programs
- A confrontation with modern computer-assisted math
- Discussion

"Quick" tour through ENIAC

"Quick" tour through ENIAC. Background (1)

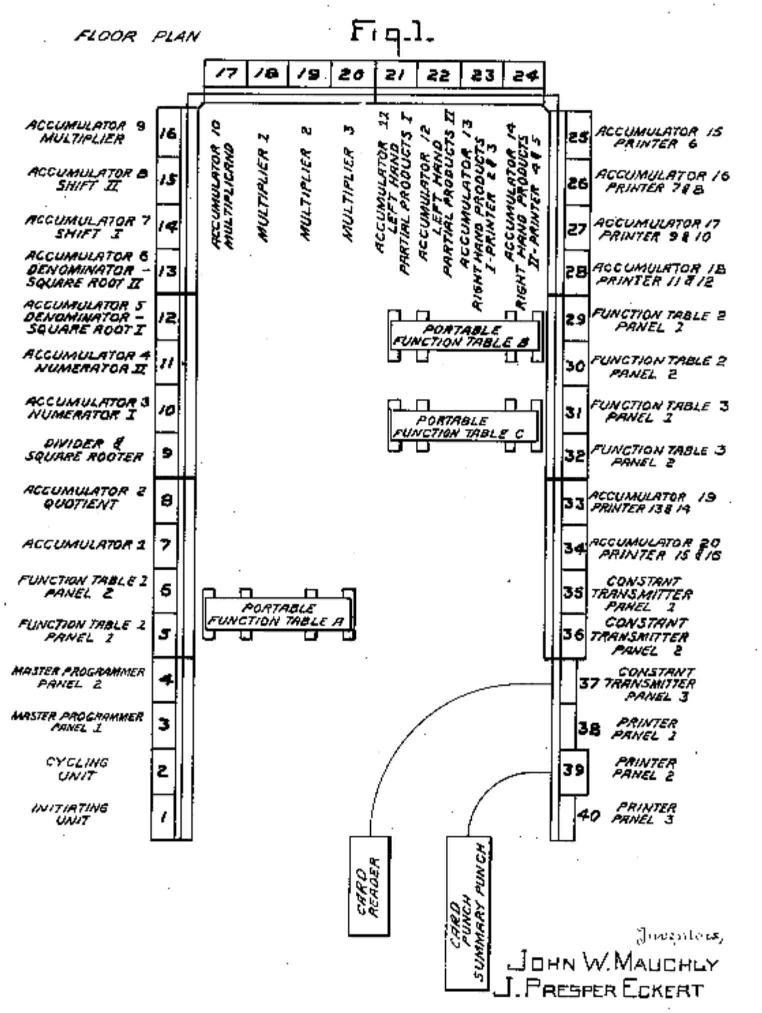
- ENIAC, The Electronic(!) Numerical Integrator And Computer
- Initial idea to build a large computer using vacuum tubes: Mauchly who wanted to predict the weather.
- In 1941, Mauchly met Presper J. Eckert at the Moore School at Penn University. Eckert "*was willing and agreeable to talk about the possibility of electronic computers [...] Nobody else really wanted to give it a second thought*" [Mauchly, 1970]

⇒ Formal proposal to the Navy Ordnance for building an electronic computer (mainly to compute firing tables). Eckert and Mauchly started building the ENIAC in 1943. Unveiled to the public on February 15, 1946

- Local and direct programming method: "The ENIAC was a son-of-a-bitch to program" (Jean Bartik)
- Initially the ENIAC was a highly parallel machine, until it was rewired in 1948

  "The original "direct programming" recabling method can best be described as analogous to the design and development of a special-purpose computer out of ENIAC component parts for each new application" (Fritz, 1994)

A "quick" tour through the ENIAC. (A. Goldstine, 1946; A.& H. Goldstine; Burks, 1981)



The units of the ENIAC.

- 20 accumulators
- a multiplier, a divider and square rooter
- a constant transmitter and 3 function tables (ENIAC's main memory storage units)
- one master programmer (a central programming unit)
- cycling unit
- initiating unit
- a card reader and a printer

Some general aspects.

- Two kinds of circuits: the *numerical* circuits for storing and processing electric signals representing numbers and *programming* circuits for controlling the communication between the different parts of the machine.
- All units had to be programmed locally, connected through program cables
- Synchronization: the central programming pulse (CPP) = one addition time = 1/5000 second.
- Each unit takes an integer number of addition times to complete its operation. If so programmed it emits a programming pulse after finishing the operation, activating the next (sub)routine.

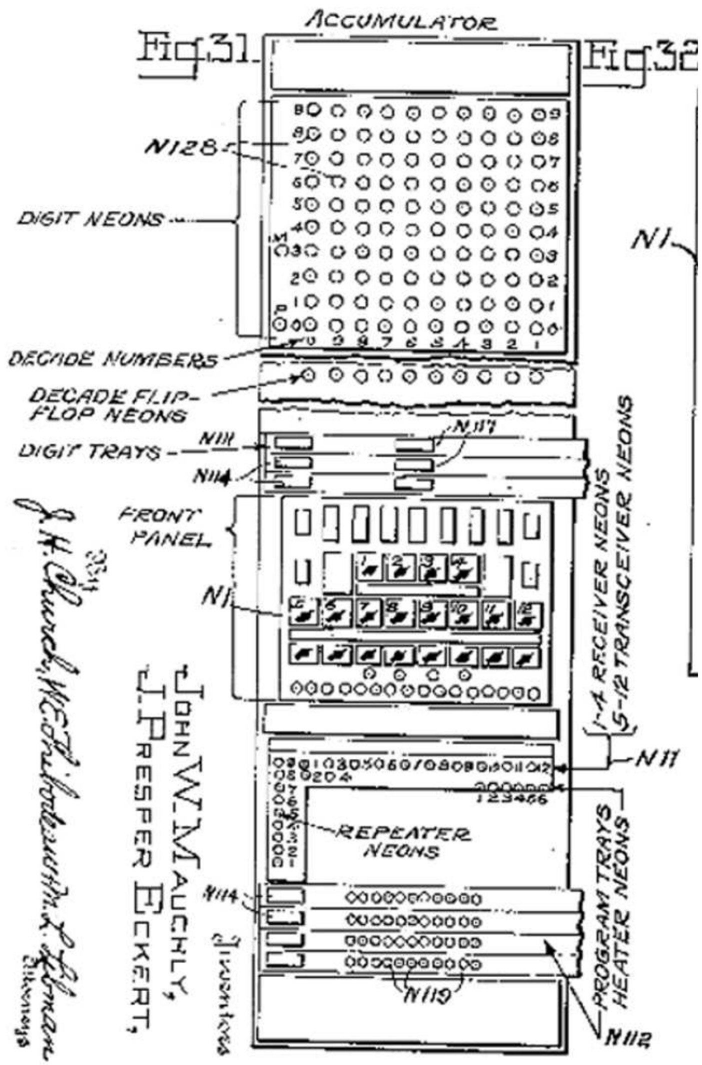
The accumulator. The main arithmetic units.

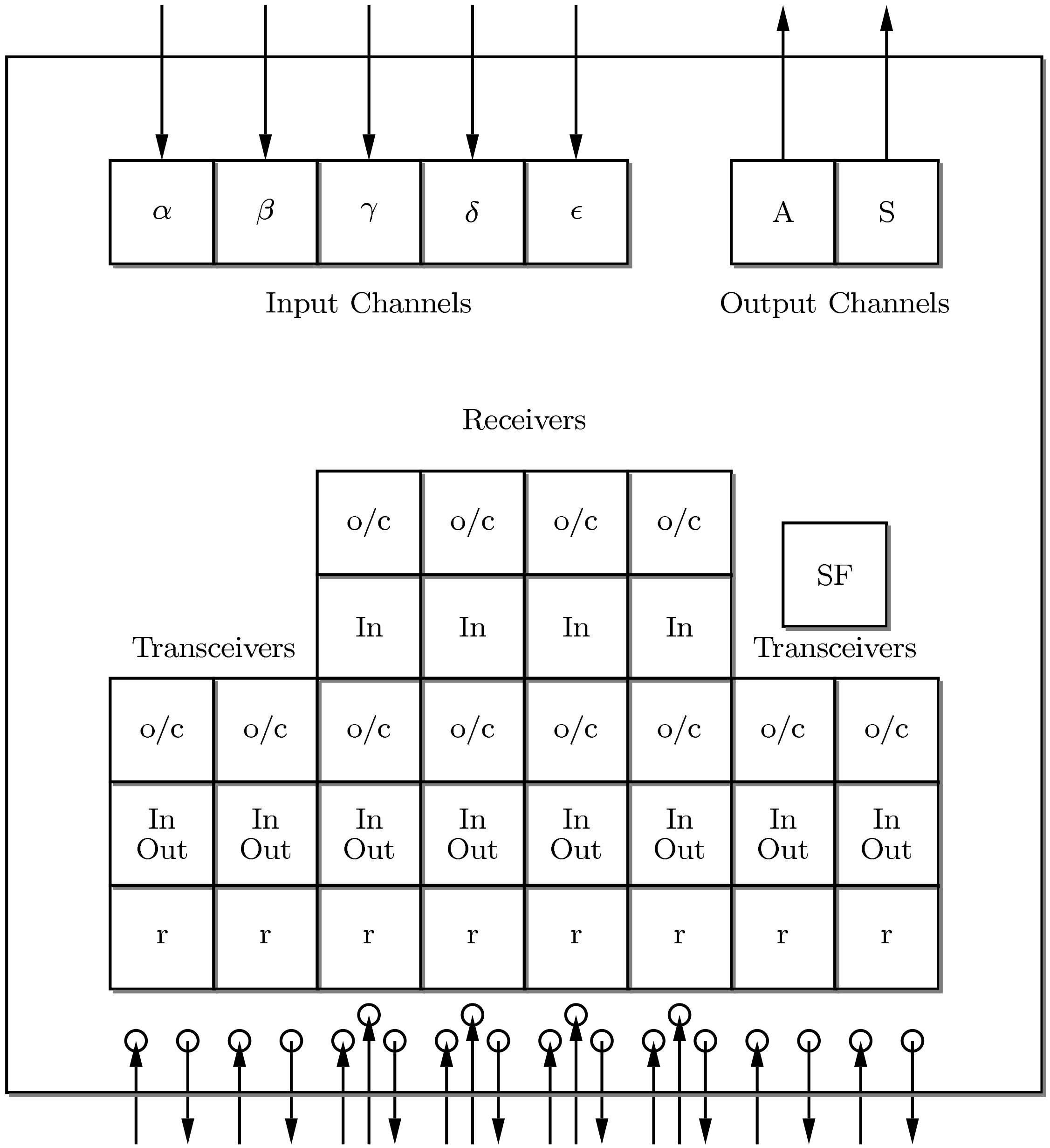
The numerical part

* Each can store a 10-decimal signed number in ten decade ring counters + PM-counter (for the sign)
* 5 input channels (α to ϵ), two output channels (A and S)to transmit a number n (through A) or its complement $10^{10} - n$ (through S), or both (AS).

The programming part

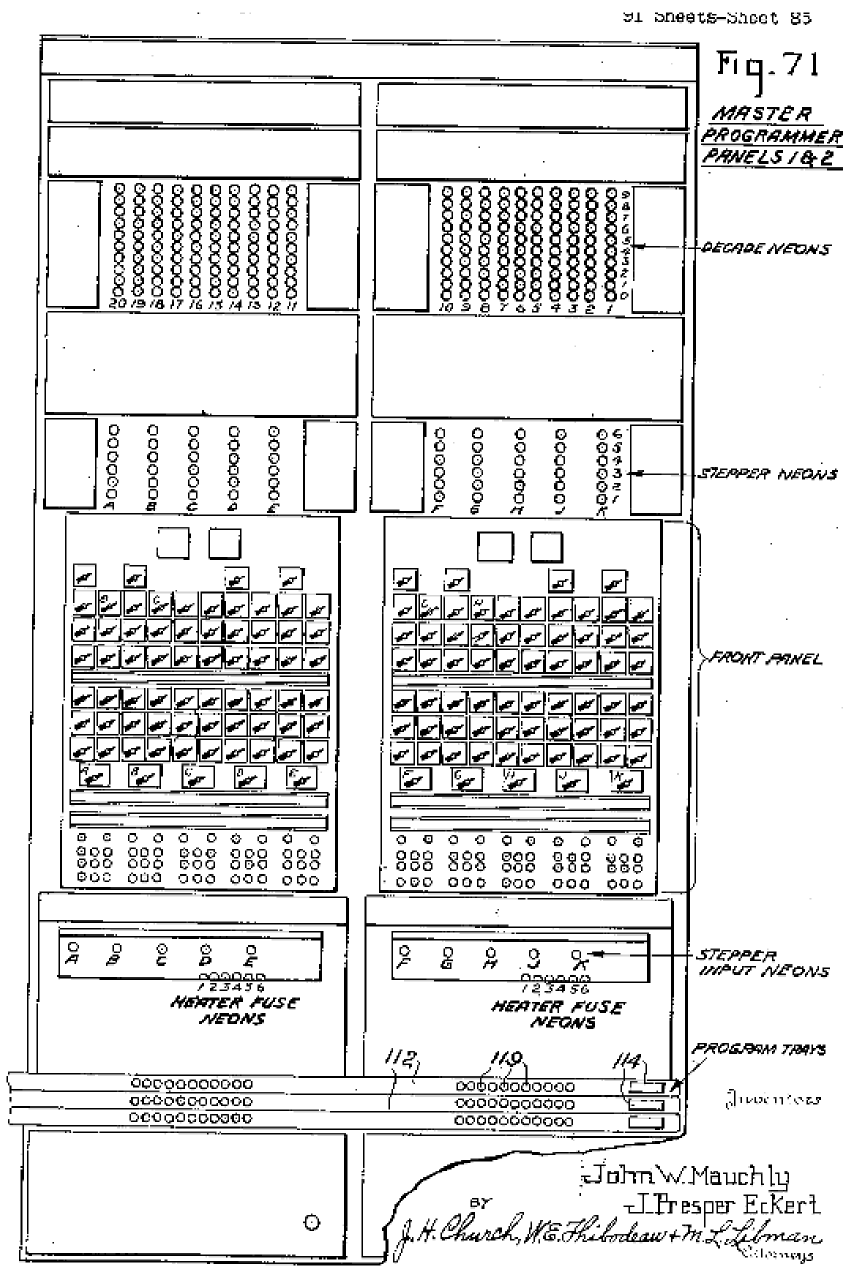
* 12 program controls: 4 receivers, 8 transceivers
* The transceiver: a program pulse input and output terminal
* The receiver: it has no program pulse output terminal and no repeater switch


ACCUMULATOR
Fig.31
Fig.32
N128
DIGIT NEONS
N1
DECADE NUMBERS
DECADE FLIP-FLOP NEONS
DIGIT TRAYS
N111
N114
N117
FRONT PANEL
1-4 RECEIVER NEONS
5-12 TRANSCEIVER NEONS
N11
REPEATER NEONS
PROGRAM TRAYS
HEATER NEONS
N119
N112
JOHN W. MAUCHLY,
J. PRESPER ECKERT,
Inventors


α
β
γ
δ
ϵ
A
S
Input Channels
Output Channels
Receivers
o/c
In
SF
Transceivers
Transceivers
o/c
In
Out
r

The master programmer. Centralized programming memory.

- 10 independently functioning units, each having a 6-stage counter (called the stepper)
- 3 input terminals for each stepper counter (the stepper input, direct input and clear input)
- 6 output terminals for each stage of the stepper. Each such stage s was associated with a fixed number d_s by manually setting decade switches, and with 1 to 5 decade counters.


91 Sheets-Sheet 85
Fig. 71
MASTER PROGRAMMER PANELS 1 & 2
DECADE NEONS
STEPPER NEONS
FRONT PANEL
STEPPER INPUT NEONS
HEATER FUSE NEONS
PROGRAM TRAYS
112
119
114
Inventors
John W. Mauchly
J. Presper Eckert
BY
J. H. Church, W. E. Thibodeau + M. L. Libman
Attorneys

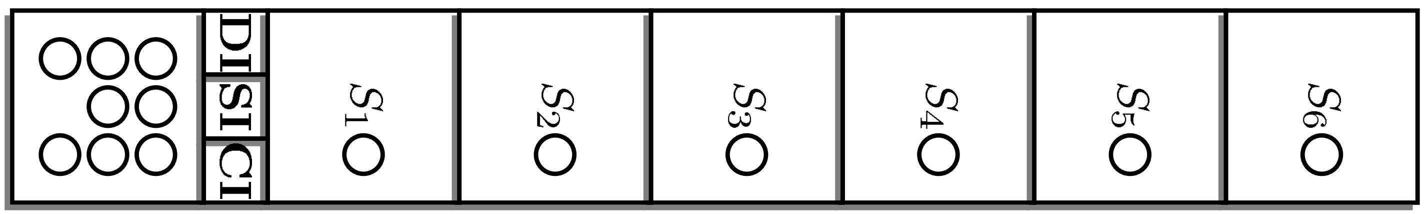


Figure 1: A Schematic (Reduced) Representation of a stepper counter of the Master Programmer.

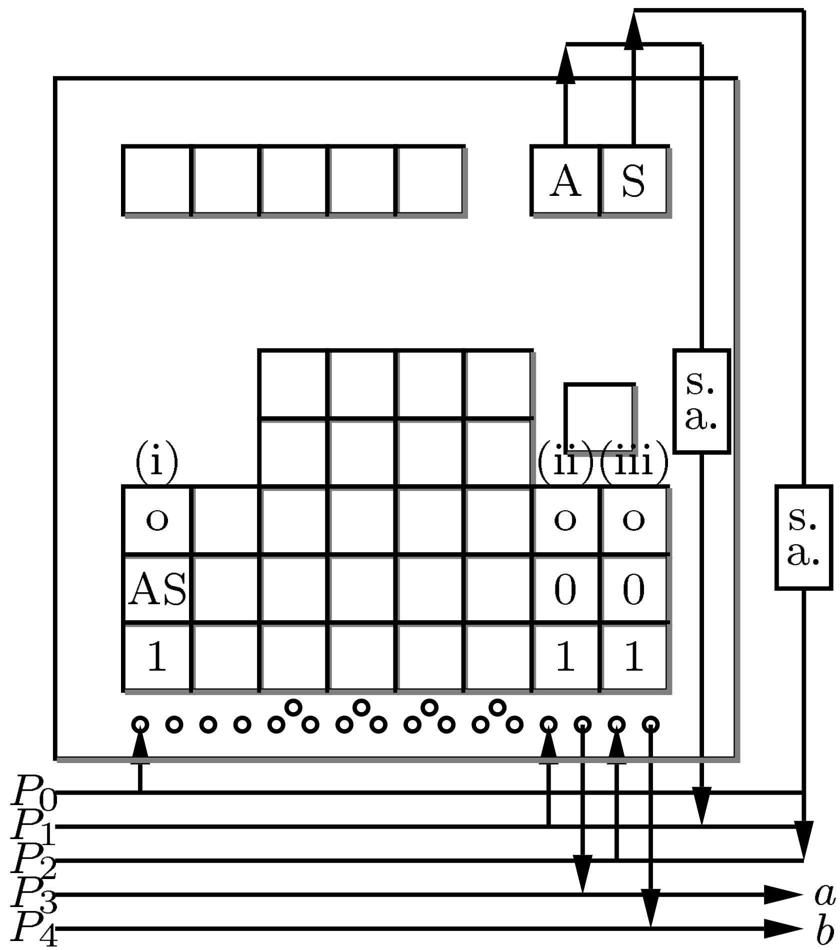
Branching...

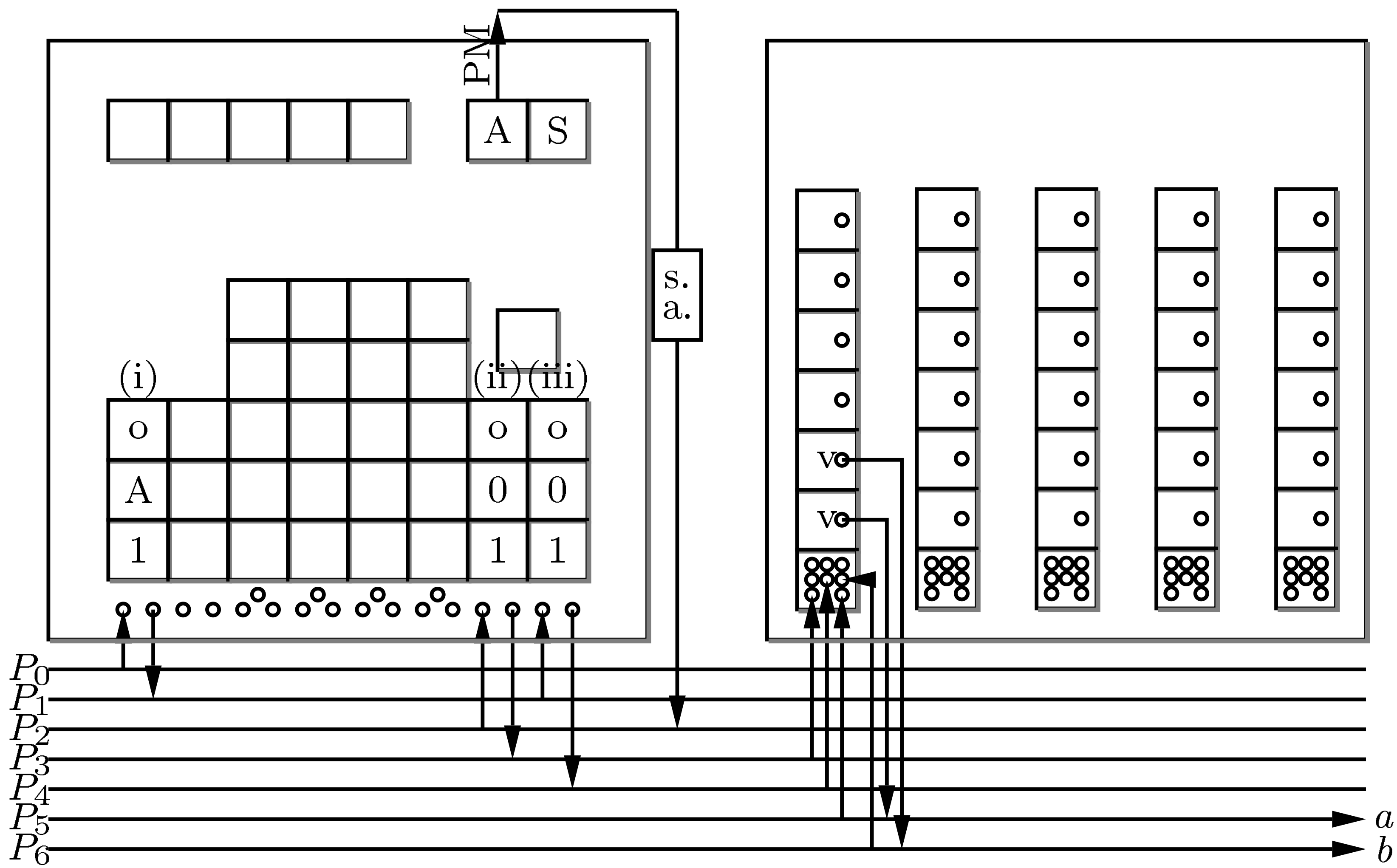
"The mathematical methods available to a computing unit depends of course on the versatility of the unit [...] The advent of large-scale computers has added a fifth operation of considerable importance, namely, discrimination. This is the operation of making a choice [...] This operation is peculiar to discrete-variable machines, since its outcome is not continuous. (Lehmer, 1951)"

- "magnitude discrimination" or "branching" : possible because 9 digit pulses were transmitted for sign indication M and none for sign indication P. The fact that digit pulses were transmitted for every digit except for 0 could be exploited in a similar manner.
- special adaptor for transforming digit pulse into programming pulse to the program pulse input terminal of an otherwise unused 'dummy (program) control'

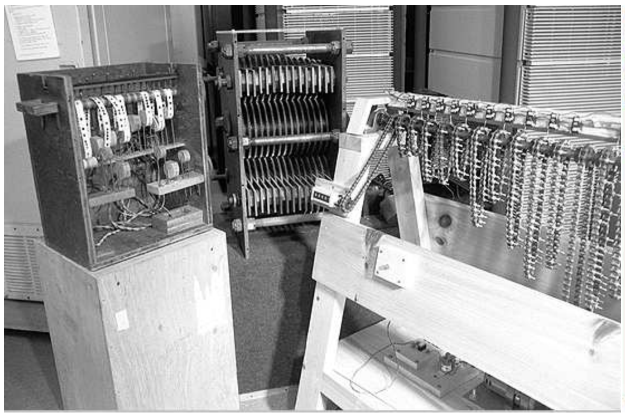
Two methods

- 'IF' with two output channels of an accumulator
- 'IF' with one output channel and a stepper


A
S
s.
a.
s.
a.
(i)
(ii)
(iii)
o
o
o
AS
0
0
1
1
1
P_0
P_1
P_2
P_3
P_4
a
b


PM
A
S
s.
a.
(i)
(ii)
(iii)
o
A
1
o
0
1
o
0
1
v
v
P_0
P_1
P_2
P_3
P_4
P_5
P_6
a
b

Lehmer, number-theory and the explorative attitude in math (Joint work with M. Bullynck)



"My father did many things to make me realize at an early age that mathematics, and especially number theory, is an experimental science." (Lehmer, 1974)

"I spent [...] two days [...] walking around in the red canyons and exploring the paleontology and archeology of the region [...] On the floor of the canyon are little postholes, and if you investigate one of these you will find a whole little world of its own, living, until it dries out of course, in this very restricted environment. That's the nature of the material I am presenting here."

How a number theorist got involved with computers...

- The Ballistic Research Laboratories (Aberdeen Proving Ground) had “assembled a ‘Computations Committee’ to prepare for utilizing the machine after its completion”, and the ENIAC was extensively test-run during its first months.
- The members:
  * Leland B. Cunningham (an astronomer)
  * Haskell B. Curry (a logician)
  * **Derrick H. Lehmer (a number theorist)**

Lehmer and the first extensive number-theoretical computation on the ENIAC

- "*[Lehmer] had programmed the problem and run it on ENIAC, with J. Mauchly serving as "computer operator", during the three-day weekend of July 4, 1946. The running time of the problem occupied almost the entire weekend, around the clock, without a single interruption or malfunction. It was the most stringent performance test applied up to that time, and would be an impressive one even today. The problem was only a "test problem" from the point of view of the Army, but* **it provided an intrinsically important result in the theory of numbers**." (Alt, 1972)
- A special (but invalid) case of the converse of Fermat's little theorem

  Theorem 1 *If n divides $2^n - 2$ then n is a prime*
- **Goal I** Testing the ENIAC
- **Goal II** Finding composite numbers to generate tables of primes

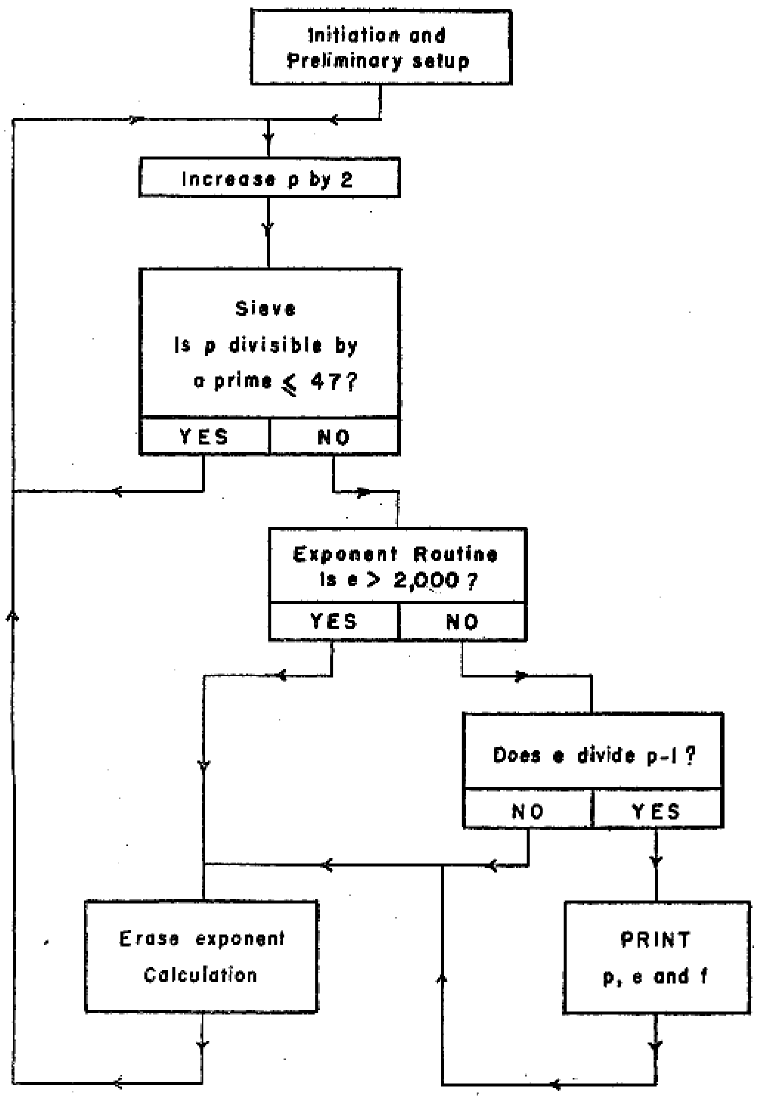
How was ENIAC used to compute composite numbers?

- The ENIAC was used to determine a list of exponents e of 2 mod p, i.e., the least value of n such that $2^n \equiv 1 \bmod p$ with p prime
- These exponents can be used to determine composite numbers of the form $2^{pq} - 2$ through the theorem:

  Theorem 2 *If p and q are odd distinct primes, then $2^{pq} - 2$ is divisible by pq if and only if p - 1 is divisible by the exponent to which 2 belongs modulo q and q - 1 is divisible by the exponent to which 2 belongs modulo p*
- Compute relatively small numbers to compute large numbers
- A sieve was implemented on the ENIAC to determine primes relative to the first 15 primes, thus making use of the ENIAC's parallelism. The last prime p processed, after 111 hours of computing time, was $p = 4538791$

TABLE OF COMPOSITE SOLUTIONS n OF FERMAT'S CONGRUENCE $2^n \equiv 2 \pmod{n}$
AND THEIR SMALLEST PRIME FACTOR p

n	p	n	p	n	p
100463443	7577	312773	3541	558011	6449
618933	4729	413333	6067	940853	503
860997	9649	495083	1987	120296677	229
907047	5023	717861	1013	517021	2341
943201	5801	111202297	5273	838609	433
101152133	5807	370141	883	121062001	1201
158093	3673	654401	6101	128361	6961
218921	8713	112032001	4001	374241	6361
270251	9001	402981	3061	121472359	4409
276579	6163	828801	6133	122166307	739
954077	1597	844131	3067	396737	2857
102004421	2381	113359321	761	941981	337 · 491
443749	4049	589601	331 · 571	123330371	691
678031	3583	605201	7537	481777	3881
690677	2069	730481	433	559837	4177
690901	5851	892589	919	671671	9631
103022551	6121	114305441	6173	886003	1187
301633	7873	329881	7561	987793	709
104078857	6679	469073	3089	124071977	2089
233141	2441	701341	1229	145473	397
524421	5903	842677	2459	793521	4561
105007549	1033	115085701	1801	818601	2281
305443	2833	174681	773	125284141	4231
919633	4603	804501	5381	686241	6473
941851	1051	873801	1051	848577	2897
106485121	7297	116090081	6221	126132553	5023
622353	433	151661	7621	886447	6793
743073	1699	321617	5393	127050067	5347
107360641	2161	617289	2357	710563	9787
543333	4889	696161	2161	128027831	11161
108596953	7369	998669	1459	079409	5437
870961	2609	117246949	1597	124151	2311
109052113	4993	445987	5419	468957	2927
231229	2699	959221	2053	536561	8017
316593	3697	987841	7681	665319	2383
437751	5231	118466401	1249	987429	4637
541461	6043	119118121	2729	129205781	6563
879837	2707	204809	2383	256273	739
110135821	3967	261113	4657	461617	10177
139499	6427	378351	911	524669	2939


Initiation and Preliminary setup
Increase p by 2
Sieve
Is p divisible by a prime ≤ 47?
YES
NO
Exponent Routine
Is e > 2,000 ?
YES
NO
Does e divide p-1 ?
NO
YES
Erase exponent Calculation
PRINT
p, e and f

Computing the exponent e: the machine's point of view

"*[The method used by the ENIAC is] based directly on the definition of* e*, namely, to compute*

$$2^n \equiv \Gamma_n (\text{mod} p), n = 1, 2, ...$$

until the value 1 appears or an until $n = 2001$*, whichever happens first. Of course, the procedure was done recursively by the algorithm:*

$$\Gamma_1 = 2, \Gamma_{n+1} = \begin{cases} \Gamma_n + \Gamma_n & \textit{if } \Gamma_n + \Gamma_n < p \\ \Gamma_n + \Gamma_n - p & \textit{otherwise} \end{cases}$$

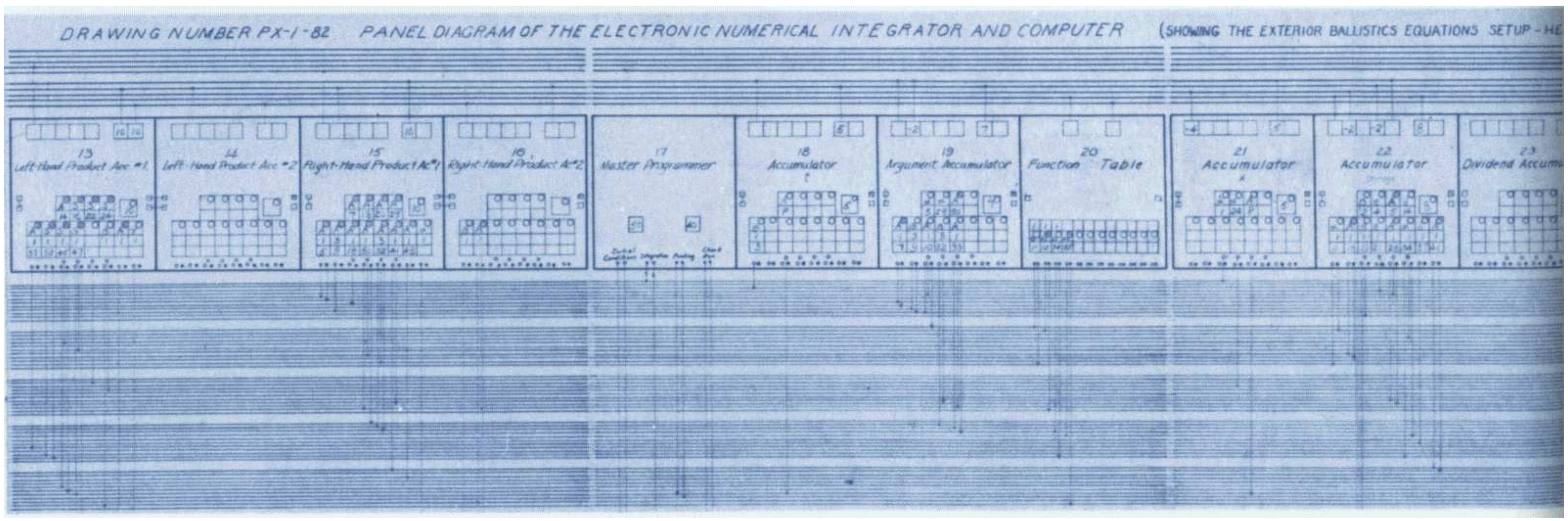
Only in the second case can Γ_{n+1} *be equal to 1. Hence this delicate exponential question in finding* e(p) *can be handled with only one addition, subtraction, and discrimination at a time cost, practically independent of* p*, of about 2 seconds per prime. This is less time than it takes to copy down the value of* p *and in those days this was sensational.*" (Lehmer, 1974)

A Prime Sieve: internalization, parallelism and heuristic programming

- making use of the ENIAC's parallelism
- Internal "call-by-value" of primes, instead of "slow" external feeding
- Minimizing the chance that $p = 2n + 1$ is not a prime relative to the primes ≤ 47, + divisibility $p - 1$ by e. Remainder (25 out of 11336) eliminated by hand
- Eratosthenes's Sieve:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & \dots \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & \dots \end{pmatrix}$$

The Reconstruction



Eniac set-up diagram.

"Well, we were happy to have a wiring diagram. On the ENIAC that was our language""

Reconstruction of the Sieve

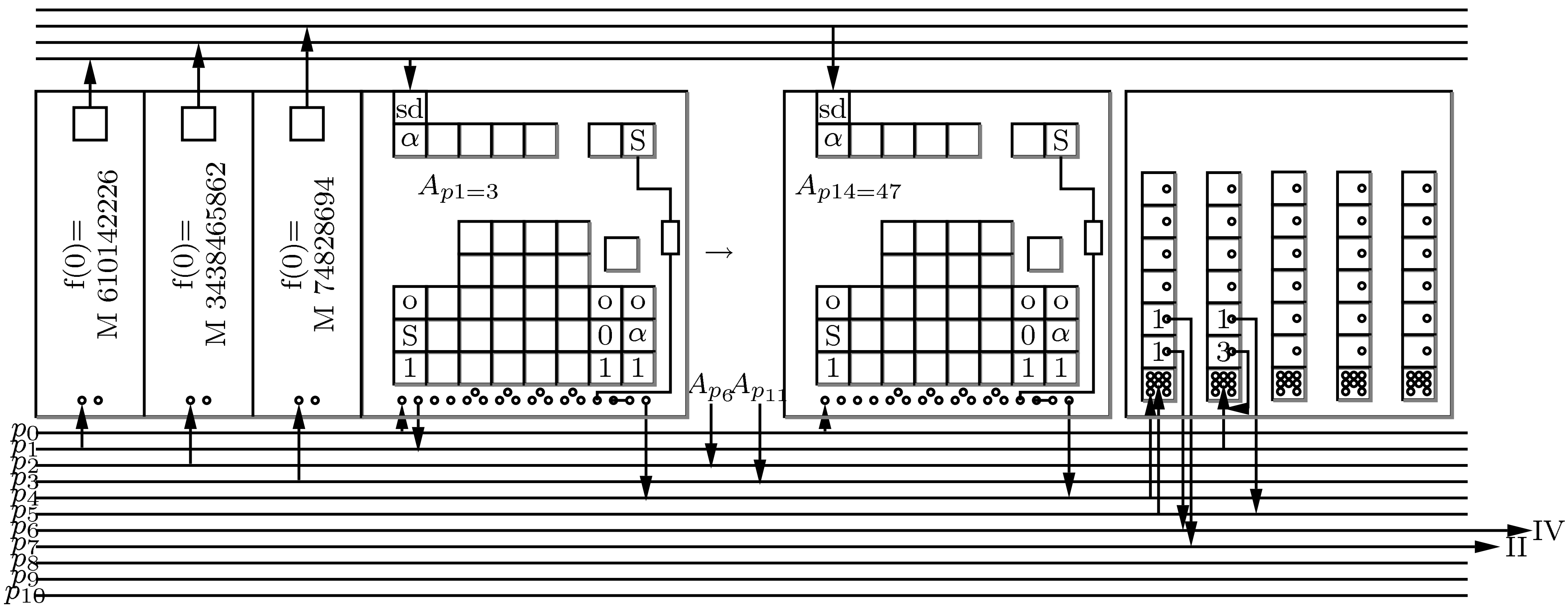
- One accumulator for each prime $2 < p_j \leq 47$, resulting in 14 accumulators for the sieve.
- Initial set-up:
  * In each accumulator A_{p_j}, set complement of $p_j - 1$, e.g. $A_{p_{14}}$ will contain M 9999999954.
  * Initiating program pulse (pp) to (a) first transceiver T1 of each A_{p_j}, operation switch set to α, plus repeater set to 1 (b) the constant transmitter. This will send the number two to each of the A_{p_j}
- The next steps: check for each A_{p_j} in parallel whether $P = 2r+1$ is divisible by p_j
  - **Checking routine**. Use of second branching method, connecting the PM lead of the S output of each of the A_{p_j} to 14 dummy controls (T2). If P is divisible by p_j, the number contained in A_{p_j} will be P 0000000000 and thus positive, while it will be negative in all other cases (this is why we use complements). If a given A_{p_j} stores P 0000000000, and P is thus divisible by p_j, A_{p_j} has to be reset to the complement of $2p_j$.
  - **The problem of loading** $2p_j$. Only those that contain P 0000000000 should receive a number (**Problem 1**) and each must receive a different number (**Problem 2**).

Reconstruction of the Sieve. Solution of the two main problems

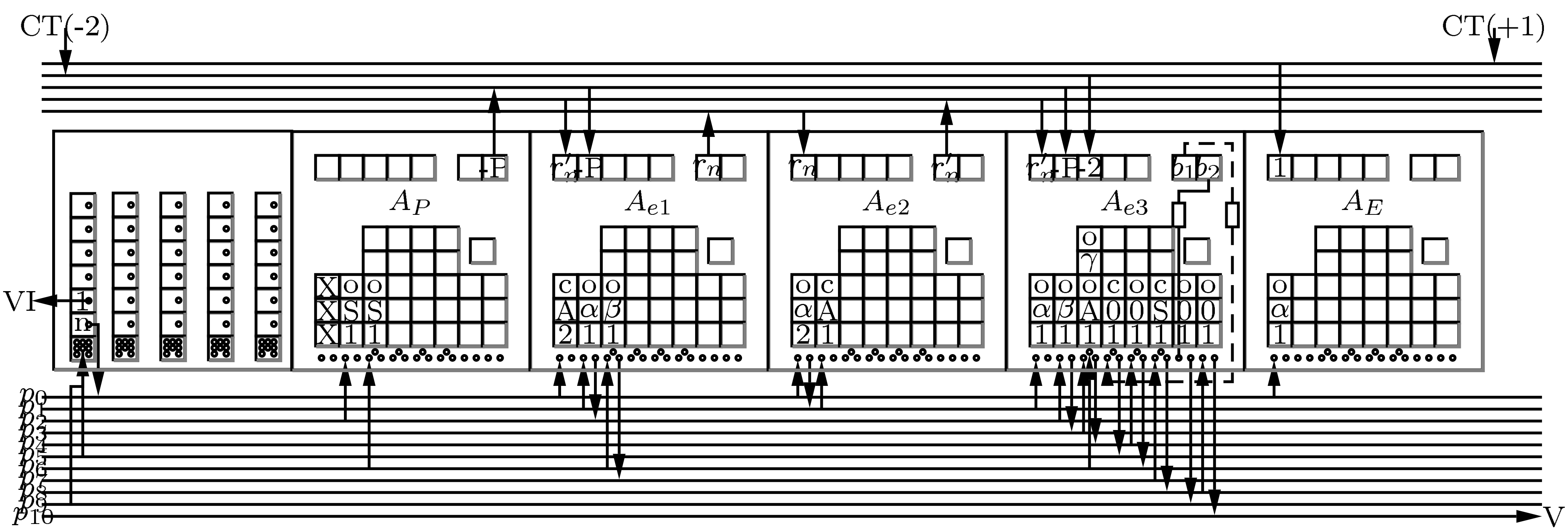
Problem 1. *Directly* connect the program pulse output terminal of each of the dummy controls (T2) of the A_{p_j} to the program pulse input terminal of one of the transceivers (T3) of each of the A_{p_j}. This could be done by using a **loaded program jumper** (A. Goldstine, 1946). Each T3 of an A_{p_j} is set to receive once through input channel α, β or γ depending on the group A_{p_j} belongs to.

Problem 2. Use of the three function tables and special digit adaptors. The 14 A_{p_j}'s are divided into three groups: A_{p_1} – A_{p_5}, A_{p_6} – $A_{p_{10}}$, $A_{p_{11}}$ – $A_{p_{14}}$. In each group, the PP output terminal of T1 of rsp. A_{p_1}, A_{p_6} and $A_{p_{11}}$ is connected to three different program cables. The first of these cables sends a PP to function table 1, the second to function table 2 and the last to function table 3. Each of the function tables contains rsp. one of the following values: M 610142226, M 3438465862 and M 64828694 at place 0 (function value $f(0)$). Each of these values will be sent through the respective input channels α, β and γ and then be converted in the correct way through an adaptor connecting a **shifter and deleter adaptors**.

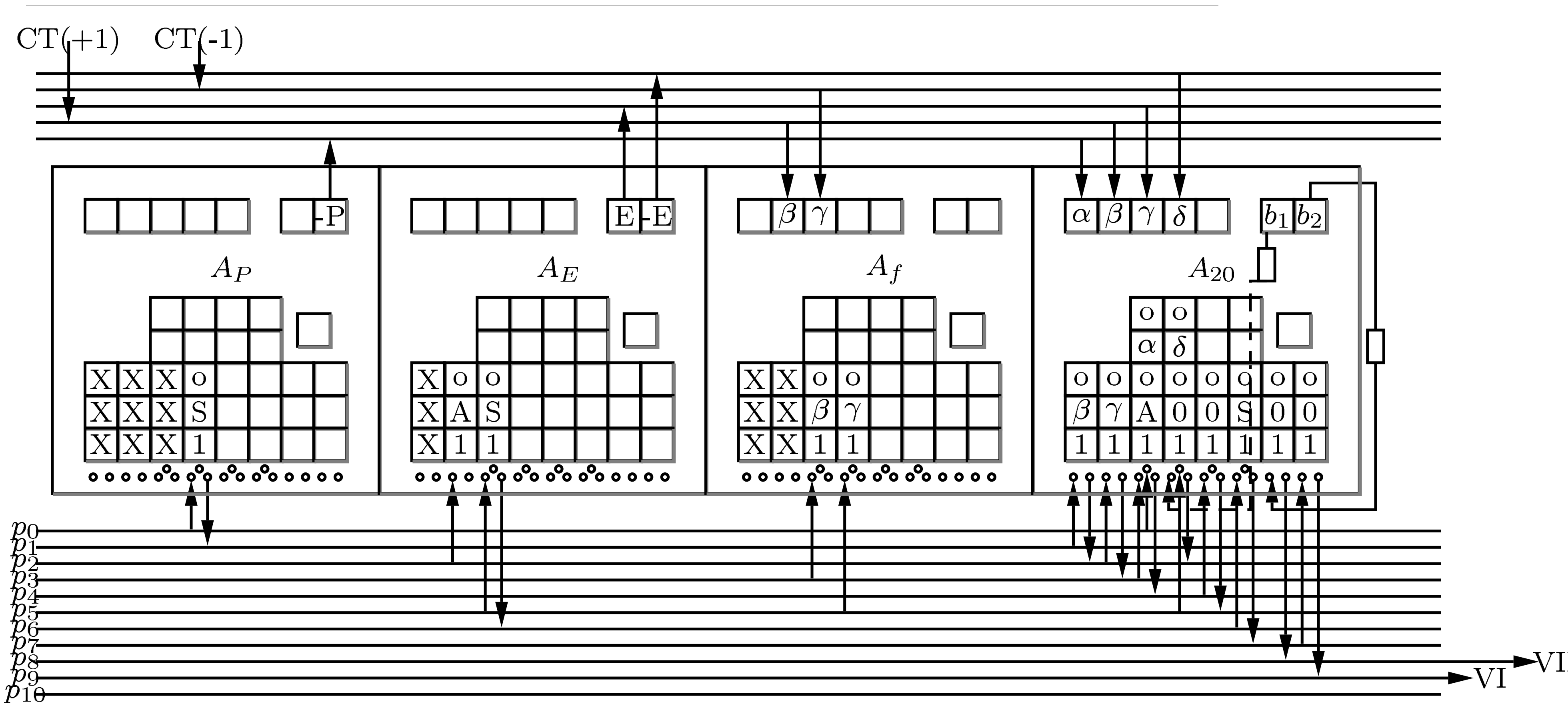
Reconstruction of the Sieve



Reconstruction of the Exponent Routine



Reconstruction of the Division Routine



Lehmer's vision on computing & math: the machine as a collaborator

- "The computer as a means to disclose the universe of mathematics: "[T]he most important influence of the machines on mathematics should lie in the opportunities that exist for applying the **experimental method** to mathematics. [...] Many a young Ph.D. student in mathematics has written his dissertation about a class of objects without ever having seen one of the objects at close range. There exists a distinct possibility that the new machines will be used in some cases to **explore** the terrain that has been staked out so freely and that something worth proving will be discovered in the rapidly expanding universe of mathematics."
- Lehmer's classification of human-machine mathematics
  - Searching for counterexamples
  - Verification and exploration of cases of a proposition to find ideas for a proof (or formulate support for conjecture)
  - Construction and inspection of tables: "Not only is the publication of such tables impossible; even the inspection is well beyond human capability. **It soon becomes apparent that it should be the machine's responsibility to make this inspection**"
  - Verification of a large number of cases $\Rightarrow$ Lehmer's version of "true" theorem proving

Lehmer's vision on computing & math: significance of "hands-on"

- **To know the machine...** *A lot of the people around here know a machine, the computing machine is a place where you leave the deck and then there is a place where you pick up the paper. That's what a computing machine is. [...] And they are fighting this machine, trying to get it to respond to their demands, finally succeeding; that's what a machine is to them. They really don't have any – I guess the way we say it today:* **they don't have a sense of identity with the machine. We used to have, when we had "hands on" policies, you know**

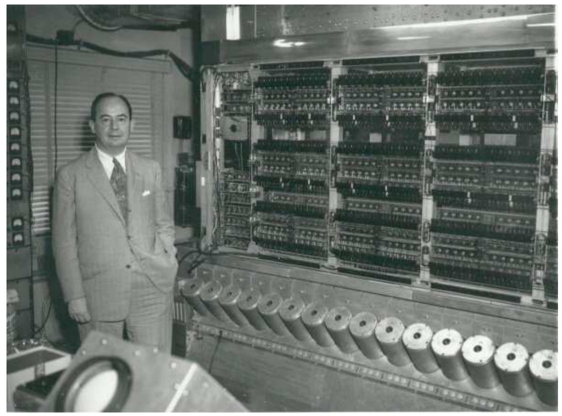
- "The **language problem** is a case in point. Languages like ALGOL and FORTRAN stand between the user and the machine to "help him communicate." [O]f course, the contemplated user is never a number theorist. For example in FORTAN II all positive integers are less than 32,768 and multiplication is only approximate [L]anguages cost real money. However, the needs of the number theorist are pretty well met by a package of much used subroutines written in machine language. (computer technology applied to the theory of numbers)"

- "As things become more and more automated, of course, it began to separate from the machines to some extent; helping it communicate, but it also is a barrier between the operator, between the user and the machine"

Lehmer's vision on computing & math: unpredictability and speed

"*In casting about for genuine theorems the proofs of which will tax the powers of a human being,* **we want to exploit the speed of the machine**. *This means that the proof must involve* **many thousands of steps** *all sufficiently different so that the* **outcome cannot be forecast**. *We must also exploit those features of the logical system of the machine that* **permit it to supervise and organize its own program. We should make it proceed in an unpredictable way by laying its own track ahead of it like a caterpillar tractor.** *At the same time it should keep a record of where it has been, so that it can return at a previous point and branch out along another path whenever it decides that this is necessary. Humans find this kind of work difficult even when it occurs in only moderate amounts.*"

"Johny", Monte Carlo, the bomb and the flow diagram

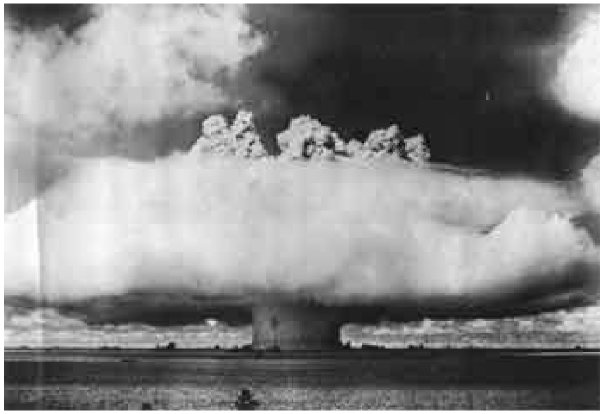


"Goldstine had met von Neumann at the Aberdeen railroad station" (Eckert, 1980)

"For a whole host of reaons [he] had become seriously interested in the thermonuclear problem being pawned at that time in Los Alamos by a friendly fellow-Hungarian scientist, Edward Teller, and his groups. Johny [...] let it be known that construction of the ENIAC was nearing completion, and he wondered whether Stan Frankel and I would be interested in preparing a preliminary computational model of a thermonuclear reaction for the ENIAC." (Metropolis)

Johny and the The Monte Carlo method

Remark dated 1983 by Ulam: "*The first thoughts and attempts I made to practice [the Monte Carlo Method] were suggested by a question which occurred to me in 1946 as I was convalescing from an illness and playing solitaires. The question was* **what are the chances that a Canfield solitaire laid out with 52 cards will come out successfully? After spending a lot of time trying to estimate them by pure combinatorial calculations, I wondered whether a more practical method than "abstract thinking" might not be to lay it out say one hundred times and simply observe and count the number of successful plays. This was already possible to envisage with the beginning of the new era of fast computers** *Later [in 1946, I] described the idea to John von Neumann and we began to plan actual calculations.*



Description of an example by von Neumann in a letter to Richtmyer as explained by Metropolis: "*The idea is* **to follow the development of a large number of individual neutron chains [A]t each stage a sequence of decisions has to be made based on statistical probabilities** *[T]he first two decisions occur at time $t = 0$, when a neutron is selected to have a certain velocity and a certain spatial position. The next decisions are the position of the first collision and the nature of that collision. If it is determined that a fission occurs, the number of emerging neutrons must be decided upon, and each of these neutrons is eventually followed in the same fashion as the first. If the collision is decreed to be a scattering, appropriate statistics are invoked to determine the new momentum of the neutron. [T]hus, a genealogical history of an individual neutron is developed. The process is repeated for other neutrons until a statistically valid picture is generated. [...]* **How are the various decisions made? To start with, the computer must have a source of uniformly distributed pseudo-random numbers.**

"Johny" and the Monte Carlo method

- exploring neutron chain reactions in fission devices: estimation of multiplication rate to predict explosive behavior fission device
- "The statistical approach is very well suited to a digital treatment"
- **Limited memory and External representation of neutrons**: "character" determined by size of punched card: "each neutron is represented by [an 80 entry punched computer] card [which carries its characteristic] that is, such things as the zone of material the neutron was in, its radial position [...] its velocity, and the time **[but also] the necessary random values**"
- **Speed** "I doubt that the processing of 100 'neutrons' will take much longer than the reading, punching and (once) sorting time of 100 cards; i.e., about 3 minutes. Hence, taking 100 'neutrons' through 100 of these stages should take about 300 minutes"
- "For each of thousands of neutrons, the variables describing the chain of events are stored, and this collection constitutes **a numerical model** of the process being studied. The collection of variables is analyzed using statistical methods identical to those used to analyze experimental observations of physical processes" (Eckhard, 1987)

⇒ Problem: random numbers.....

⇒ What kind of random number generator was used?

[Metropolis] suggested an obvious name for the statistical method – a suggestion not unrelated to the fact that Stan had an uncle who would borrow money from relatives because he "just had to go to Monte Carlo."

- The "quadratic" iterator: $x_n = 4x_{n-1}(x_{n-1} - 1)^2 \rightarrow$ "Any physically existing machine has a certain limit of precision. [I]n each transformation any error will be amplified on the average by approximately two. In about 33 steps the first round-off error will have grown to about 10^{10}. No matter how random the sequence is in theory, after about 33 steps one is really only testing the random properties of the round-off error. Then one might as well admit that one can prove nothing, because the amount of theoretical information about the statistical properties of the round-off mechanism is nil"

- **Von Neumann's middle square method (used on ENIAC):** Take a number x_0, of length n, square it, resulting in y_0 of length $2n$, extract the middle n-digits, resulting in a new number x_1, square it, resulting in y_1, ...
  $\Rightarrow$ "it is seen that this process cannot be recommended as a source of random digits. " (Lehmer, 1951)

"Johny", ENIAC and the randomness of π and e. Context

Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin. von Neumann, 1949

- "Early in June, 1949, Professor John von Neumann expressed an interest in the possibility that the ENIAC might sometime be employed to determine the value of π and e to many decimal places with a view toward obtaining a statistical measure of the randomness of distribution of the digits [...] Further interest in the project on π was expressed in July by Dr. Nicholas Metropolis [...]" (Reitwiesner, 1950)

π = 3.14159	26535	89793	23846	26433	83279	50288	41971	69399	37510
58209	74944	59230	78164	06286	20899	86280	34825	34211	70679
82148	08651	32823	06647	09384	46095	50582	23172	53594	08128
48111	74502	84102	70193	85211	05559	64462	29489	54930	38196
44288	10975	66593	34461	28475	64823	37867	83165	27120	19091
45648	56692	34603	48610	45432	66482	13393	60726	02491	41273
72458	70066	06315	58817	48815	20920	96282	92540	91715	36436
78925	90360	01133	05305	48820	46652	13841	46951	94151	16094
33057	27036	57595	91953	09218	61173	81932	61179	31051	18548
07446	23799	62749	56735	18857	52724	89122	79381	83011	94912
98336	73362	44065	66430	86021	39494	63952	24737	19070	21798
60943	70277	05392	17176	29317	67523	84674	81846	76694	05132
00056	81271	45263	56082	77857	71342	75778	96091	73637	17872
14684	40901	22495	34301	46549	58537	10507	92279	68925	89235
42019	95611	21290	21960	86403	44181	59813	62977	47713	09960
51870	72113	49999	99837	29780	49951	05973	17328	16096	31859
50244	59455	34690	83026	42522	30825	33446	85035	26193	11881
71010	00313	78387	52886	58753	32083	81420	61717	76691	47303
59825	34904	28755	46873	11595	62863	88235	37875	93751	95778
18577	80532	17122	68066	13001	92787	66111	95909	21642	01989
38095	25720	10654	85863	27886	59361	53381	82796	82303	01952
03530	18529	68995	77362	25994	13891	24972	17752	83479	13151
55748	57242	45415	06959	50829	53311	68617	27855	88907	50983
81754	63746	49393	19255	06040	09277	01671	13900	98488	24012
85836	16035	63707	66010	47101	81942	95559	61989	46767	83744
94482	55379	77472	68471	04047	53464	62080	46684	25906	94912
93313	67702	89891	52104	75216	20569	66024	05803	81501	93511
25338	24300	35587	64024	74964	73263	91419	92726	04269	92279
67823	54781	63600	93417	21641	21992	45863	15030	28618	29745
55706	74983	85054	94588	58692	69956	90927	21079	75093	02955
32116	53449	87202	75596	02364	80665	49911	98818	34797	75356
63698	07426	54252	78625	51818	41757	46728	90977	77279	38000
81647	06001	61452	49192	17321	72147	72350	14144	19735	68548
16136	11573	52552	13347	57418	49468	43852	33239	07394	14333
45477	62416	86251	89835	69485	56209	92192	22184	27255	02542
56887	67179	04946	01653	46680	49886	27232	79178	60857	84383
82796	79766	81454	10095	38837	86360	95068	00642	25125	20511
73929	84896	08412	84886	26945	60424	19652	85022	21066	11863
06744	27862	20391	94945	04712	37137	86960	95636	43719	17287
46776	46575	73962	41389	08658	32645	99581	33904	78027	59009
94657	64078	95126	94683	98352	59570	98258			

Figure 2: The first 2035 digits of π computed by the ENIAC, at the Ballistics Research Laboratory.

"Johny", ENIAC and the randomness of π and e

- **Possibility of error**: "In order to insure absolute digital accuracy, the programming was arranged so that one half applied to computation and the other half to checking. Before any deck of cards was employed to determine the next i digits, the cards were reversed and employed in the checking sequence to each division by a multiplication and each adition by a subtraction and vice versa [...]"
  The ENIAC determinations of both π and e confirm the [previously made] 808-place determination[s] of e and π
- **Time issues** The computation of π was completed over the labor-day week end through the combined efforts of four members of the ENIAC staff [...] Fritz and the author, taking turns on eight-hour shifts to keep the ENIAC operating continuously throughout the week end."
- **External/human exploration**

Planning and coding of problems for an electronic computing instrument (with H. Goldstine)

- "Since **coding is not a static process of translation**, but rather the technique of providing a dynamic background to control the automatic evolution of a meaning, it has to be viewed as a logical problem and one that represents a new branch of formal logics."

- "It is advisable [...] to plan first the course of the process and the relationship of its successive stages to their changing codes, and to extract from this the original codes sequence as a secondary operation [...] We therefore propose to begin the planning of a coded sequence by laying out a schematic of the course of C through that sequence. [T]his schematic is **the flow diagram** of C. [T]o make the flow diagram of C the first step in code-planning appears to be extensively **justified by our own experience** with the coding of actual problems.

- Composition of programs? "This possibility should, more than anything else, remove a bottleneck at the preparing, setting up and coding of a problem, which might otherwise be quite dangerous."
  BUT: the "preparatory routine" does only one thing: changing the location numbers in the subroutine

- "the problem of coding routines need not and should not be a dominant difficulty"

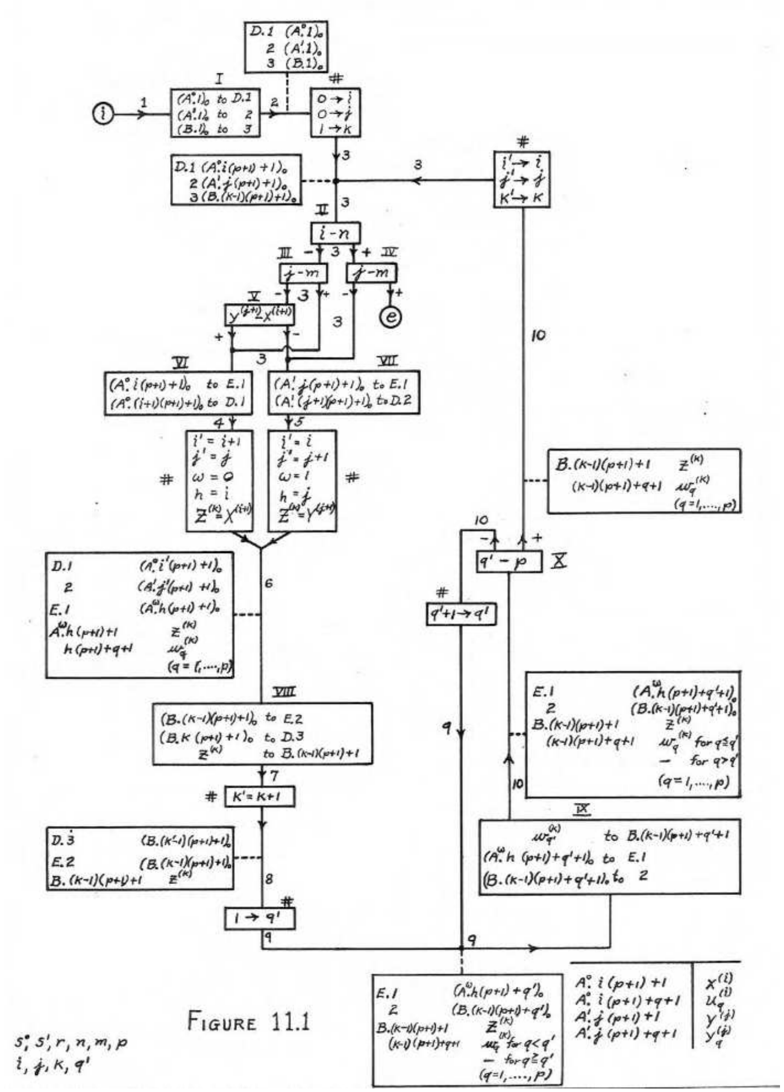


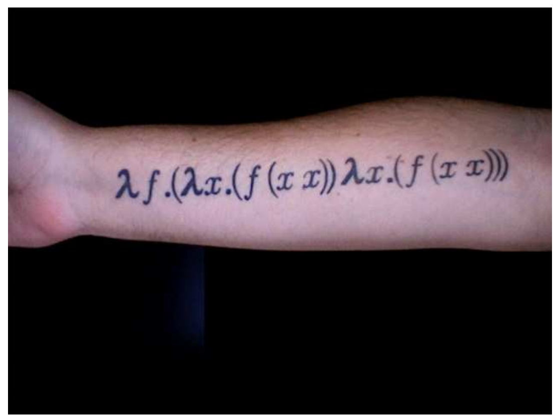
Figure 3: Flow chart for sorting

Von Neumann's vision on computing in math

- "In pure mathematics the really powerful methods are only effective when one already has some intuitive connection with the subject, when one already has [...] some intuitive insight [T]here are large areas in pure mathematics where we are blocked by a peculiar inter-relation of rigor and intuitive insight, each of which is needed for the other, and where the unmathematical process of experimentation with physical problems has produced almost the only progress which has been made. **Computing, which is not too mathematical either in the traditional sense but is still closer to the central area of mathematics than this sort of experimentation is, might be a more flexible and more adequate tool** in these areas than experimentation"

- "let me point out that we will **probably not want to produce vast amounts of numerical material with computing machines**, for example, enormous tables of functions. The reason for using fast computing machines is not that you want to pro- duce a lot of information. [...] The **really difficult problems are of such a nature that the number of data which enter is quite small. All you may want to know is a few numbers**, which give a rough curve, or one number. All you may want in fact is a yes or a no,"

Curry and the composition of programs (Joint work with M. Bullynck and M. Carlé)

Curry as the logician of ENIAC's computation committee

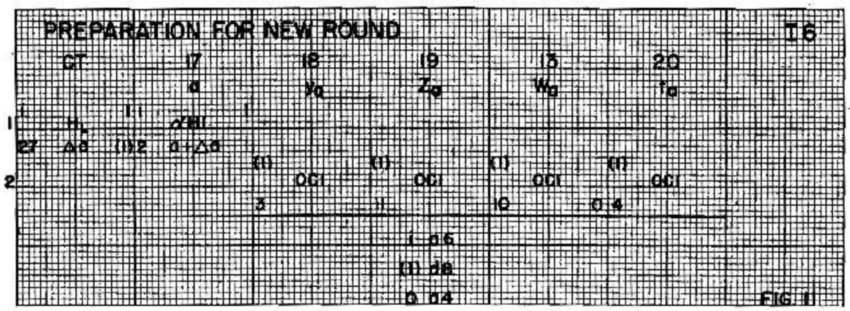
Curry and Wyatt's program on the ENIAC

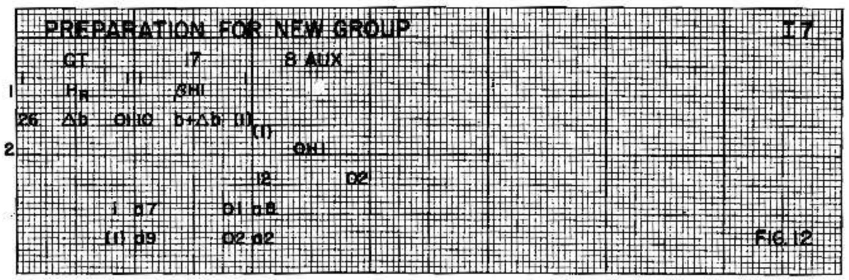
- In collaboration with Willa Wyatt, one of ENIAC's female programmers, Curry wrote up a technical report "A study of inverse interpolation of the Eniac" (1946, declassified in 1999)

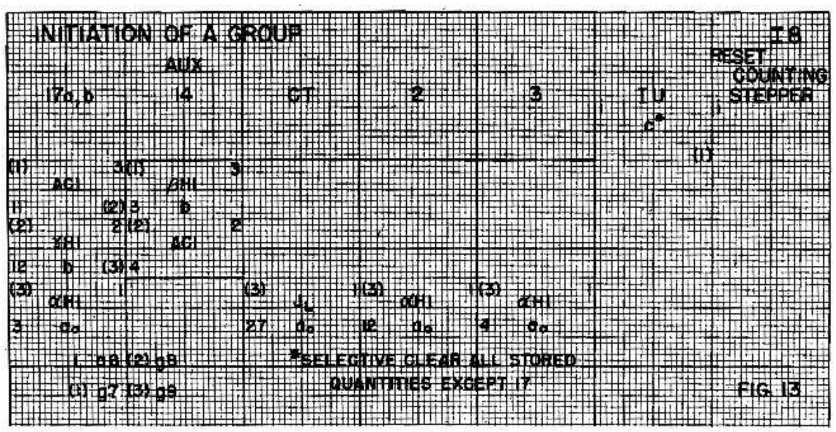

- *"The problem of inverse interpolation [...] is important in the calculation of firing tables. Suppose the trajectory calculations have given us the co-ordinates (x, y) of the projectile as functions of t (time) and ϕ (angle of departure). For the tables we want t and ϕ as functions of x and y; indeed we wish to determine ϕ so as to hit a target whose position (x, y) is known, and t is needed for the fuze setting or other purposes. [...] In this report the problem of inverse interpolation is studied with reference to the programming on the ENIAC as a problem in its own right."*

Theoretical considerations in the 1946 report

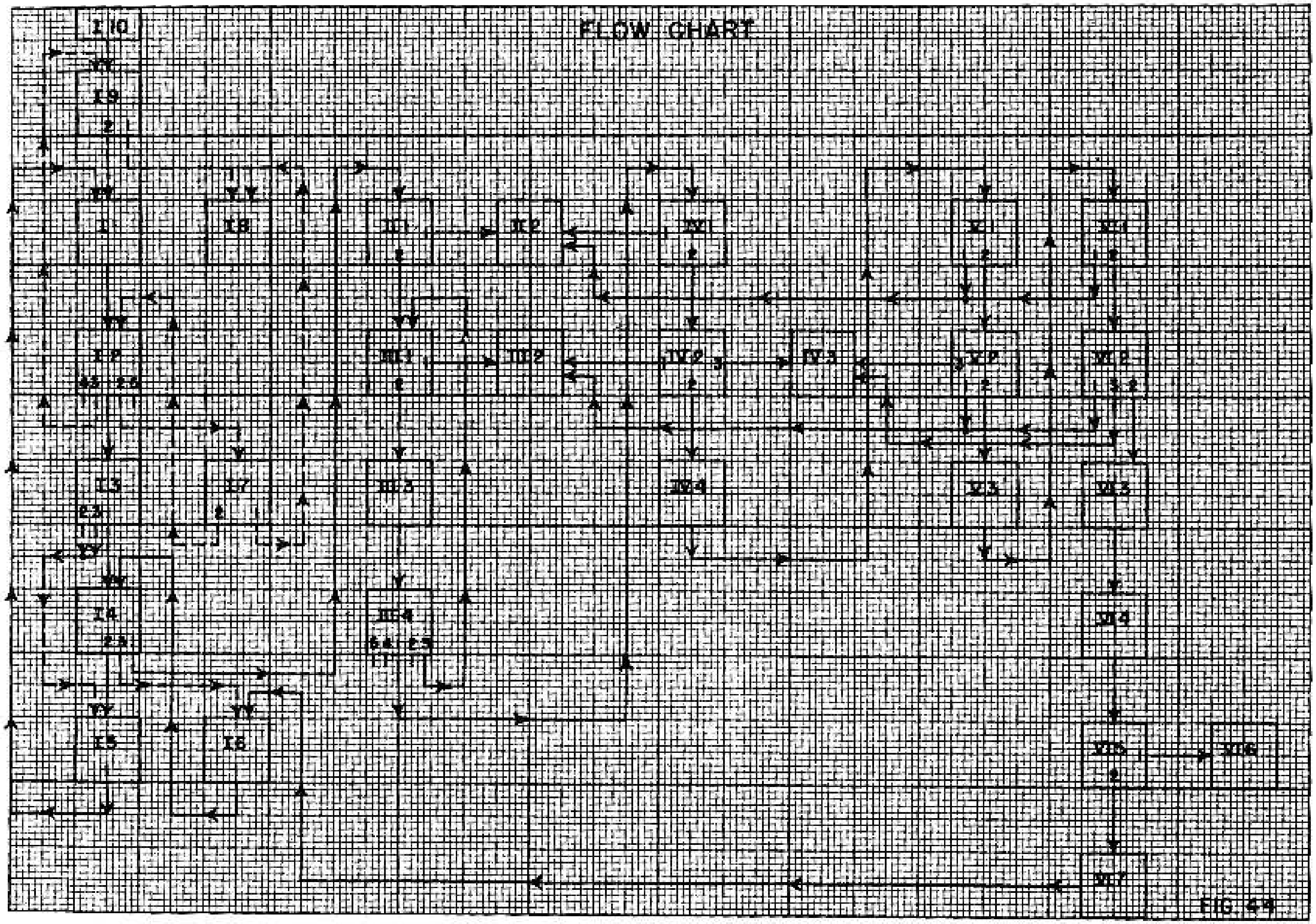
- **Stages and processes** "The stages can be programmed as independent units, with a uniform notation as to program lines, and then put together; and since each stage uses only a relatively small amount of the equipment the programming can be done on sheets of paper of ordinary size."
- The Eniac experience and the program of inverse interpolation triggers Curry's interest to develop the topic further:
  - *"The problem of program composition was a major consideration in a study of inverse interpolation on the ENIAC [...]; for although that study was made under stress and was directed primarily towards finding at least one practical method of programming a specific problem, yet an* ***effort was made to construct the program by piecing together subprograms in such a way that modifications could be introduced by changing these subprograms.****"* (Curry, 1950)
  - *"This problem is almost ideal for the study of programming; because,* ***although it is simple enough to be examined in detail by hand methods; yet it is complex enough to contain a variety of kinds of program compositions.****"* (Curry 1952)


PREPARATION FOR NEW ROUND
I6
CT
17
18
19
13
20
FIG 11


PREPARATION FOR NEW GROUP
I7
CT
17
8 AUX
FIG 12


INITIATION OF A GROUP
I8
AUX
14
CT
2
3
RESET
COUNTING
STEPPER
*SELECTIVE CLEAR ALL STORED
QUANTITIES EXCEPT 17
FIG 13

Stage	Figure	Input	No. of Program Lines		Program lines	Jumper
			Jumper	Trays		
I 2	7a	a2		12	e9,j7-j10,k7-k10, h8-h10	
I 3	8	a3	1	5	m6-m10	q8
I 4	9a	a4	1	5	m1-m5	q7
I 5	10	a5		12	1- 10,c9,c10	
I 6	11	a6		1	d8	
I 7	12	a7		1	d9	
I 8	13	a8		3	g7-g9	
I 9	14	a9		0	---	
I 10		a10				
II 1	15	b1	1	3	f1-f3	p1
II 2	15	b2		0	---	
III 1	16	b3	3	3	f4-f6	p2-p4
III 2	17	b4		5	e1-e5	
III 3	18	b5		1	b10	
III 4	19	b6		8	n1-n8	
III 5		b7			---	
III 6		b8			---	
III 7		b9			---	
IV 1	20	c1	1	3	g1-g3	q4
IV 2	16	c2	3	3	g4-g6	p5-p7
IV 3	21	c3	1	4	c4,e6-e8	q6
IV 4	22	c5		0	---	
V 1	23	c6		4	h1-h4	
V 2	16	c7	3	3	h5-h7	p8-p10
V 3	22	c8		0	---	
VI 1	23	d1	1	3	j1-j3	q5
VI 2	16	d2	3	3	j4-j6	q1-q3
VI 3	22	d3		0	---	
VI 4	24	d4		4	f7-f10	
VI 5	25	d5		5	k1-k5	
VI 6	26	d6		1	k6	
VI 7		d7			---	


FLOW CHART
FIG 44

After the ENIAC experience

- Curry reads the John von Neumann - H.H. Goldstine reports
  - *Preliminary discussion of the logical design of an electronic computing instrument.* 1946–1947 (idealized IAS machine – stored program computer)
  - *Planning and coding of problems for an electronic computing instrument.* parts I,II and III, 1947–48.
- Building upon his readings and his ENIAC experience, Curry writes up two technical reports for the Navy Ordnance (unclassified)
  - 1949: "On the composition of programs for automatic computing"
  - 1950: "A program composition technique as applied to inverse interpolation"
  - 1954: "The logic of program composition", presented at 2e Colloque International de Logique Mathématique, Paris, 25-30 août 1952 (= a short resumé of the two preceding reports)

On the composition of programs for automatic computing

- **The problem of composition**: *"In the present state of development of automatic digital computing machinery,* **a principal bottleneck is the planning of the computation***...The present report is an attack on this problem from the standpoint of composition of computing schedules. By this is meant the following. Suppose that we wish to perform a computation which is a complex of simple processes that have already been planned. Suppose that for each of these component processes we have a plan recorded in the form of what is here called a program, by means of a system of symbolization called a code. It is required to form a program for the composite computation. This problem is here attacked theoretically by using techniques similar to those used in some phases of mathematical logic."*
- **New notation and introduction of automated composition**: *"The present theory develops in fact a notation for program construction which is more compact than the "flow charts" of [Goldstine and Von Neumann]. Flow charts will be used [...] primarily as an expository device. By means of this notation a composite program can be exhibited as a function of its components in such a way that the actual formation of the composite program can be carried out by a suitable machine."*

On the composition of programs: Definitions and assumptions based on IAS machine

Program: "An assignment of $n+1$ words to the first $n+1$ locations will be called a program." $X = M_0 M_1 ... M_n$

Two types of Words: **quantities** and **orders**. Orders consist of: 1) datum number location, 2) exit number location and 3) an operator.
Mixed arithmetic order: arithmetical operation involving an order as datum (cfr. partial substition in Goldstine-von Neumann)

"The distinction between quantities and orders is not a distinction of form [...] The machine makes this distinction according to the situation. Making this classification of words in advance is a difficult problem [...] [T]he first stage in a study of programming is to impose restrictions on programs in order that the words in all the configurations of the resulting calculation can be uniquely classified into orders and quantities"

Regular program: a primary program or one that satisfies the table condition; typically determinate (restriction on the assignment of types); calculation terminates

Normal Program: $X = AC$, A is an order program and C a quantity program

On the composition of programs: transformations

Given the programs X, Y, Z and numerical function T(k):

$$\begin{array}{lcll} X & = & M_0 M_1 M_2 ... M_p & \\ Y & = & N_0 N_1 N_2 ... N_q & \\ Z & = & L_0 L_1 L_2 ... L_r & \\ T(k) & = & k' & k \leq m, k' \leq n \end{array}$$

Transformation of the first kind: changing the location numbers in a program

Y=(T)(X): T(X) gives the Y such that $n = m$ (m is range of location numbers in X) and every N_i is derived from M_i by replacing every location number k in every order of X by $T(k)$

Transformation of the second kind: reshuffling the words to match up with the changes in location numbers.

$$\{T\}(X) = Y = \begin{cases} N_0 = M_0 \\ N_{T(i)} = M_i \text{ if T is defined for i }, i > 0 \quad (*) \\ N_i = J \text{ else} \end{cases}$$

On the composition of programs: Replacement

Replacement a program made up from two programs by putting, in certain locations of one program, words from corresponding locations in the other program.

Let $\Theta \subset \{0, 1, 2, ..., p\}$ (a list of integers), then the replacement $\frac{\Theta}{Y}X = Z$ is:

$$L_i = \begin{cases} M_i \text{ if } i \notin \Theta, i \leq p \\ N_i \text{ if } i \leq q \text{ and } i \in \phi \text{ or } i > p \\ J \text{ if } i \in \Theta, i > q \end{cases}$$

When $\Theta = \emptyset$ then $\frac{X}{Y} = X$ with spaces after

On the composition of programs: Substitution

Simple **Substitution**: "A program Z will be said to be formed by substitution of Y for a certain output in X, when Z carries on a calculation homomorphic to X until the control reaches that output, then starts a calculation homomorphic to Y using the quantities calculated by X as quantity program"

Notation: $Z = X \rightarrow Y$

$X = AC$ and $Y = BC$ are normal, m is the location number ($m \in A$) at which Y ist to be substituted, then $Z = X \rightarrow Y = S_Y(X) = [\frac{\Theta T_1}{[T_2](Y)}](X) = \frac{\Theta T_1}{[T_2](Y)}(T_1)(X)$ is defined by

$$T_1(k) = \begin{cases} k & \text{for } 0 < k < m \\ m + |B| - 1 & \text{for } k = m \\ k + |B| - 1 & \text{for } m < k \leq |A| + |C| \end{cases}$$

$$T_2(k) = \begin{cases} m + k - n & \text{for } n \leq k \leq n + |B| - 1 \\ |A| + k - n & \text{for } n + |B| < k \leq n + |B| + |C| - 1 \end{cases}$$

On the composition of programs: Substitution

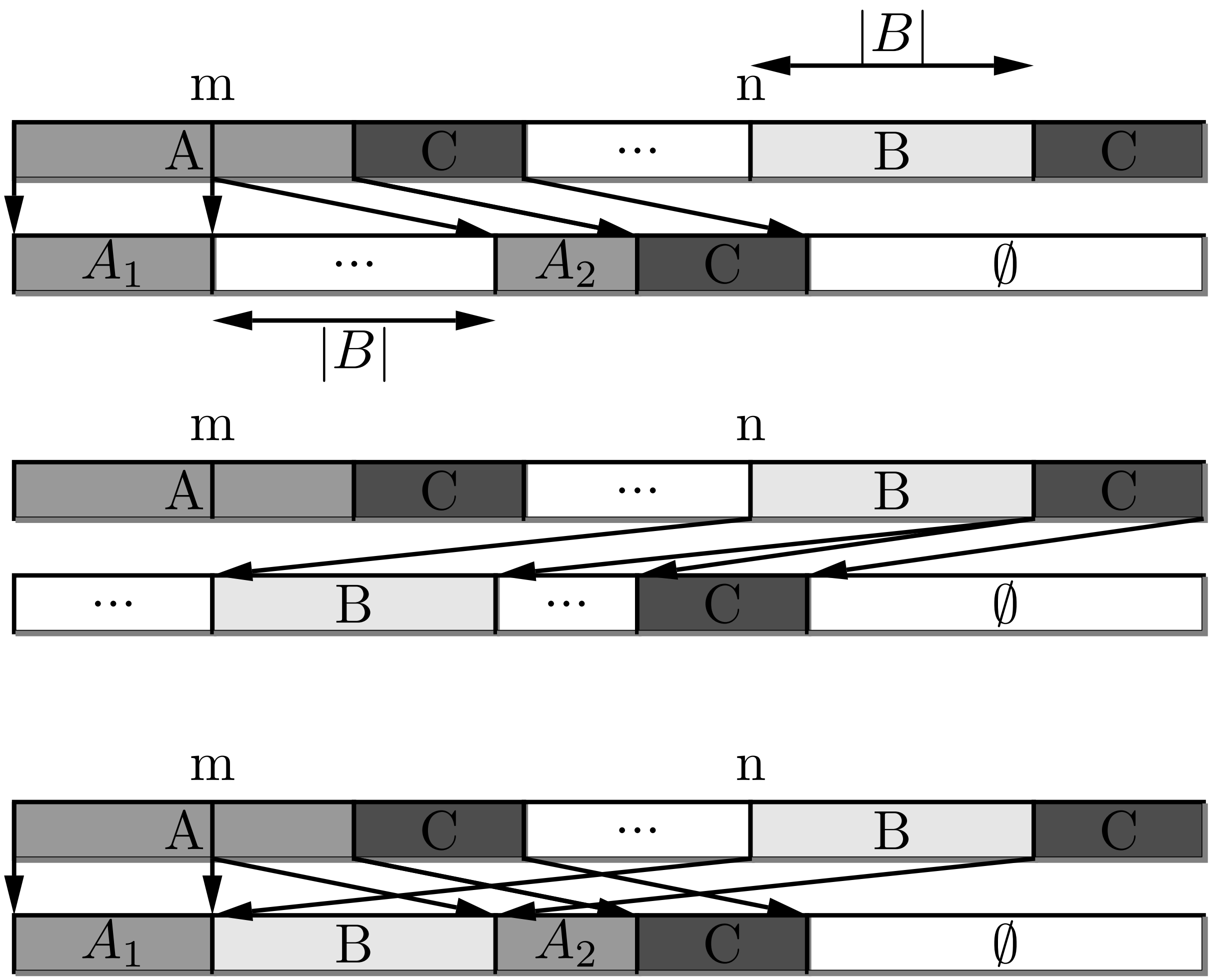


Figure 4: From top to bottom: The $T_1(X)$ transformation; the $T_2(Y)$ transformation; and finally the substitution $[\frac{\Theta T_1}{[T_2](Y)}](X)$ that substitutes Y in X at position m.

1949: "On the composition of programs for automatic computing"

Notations...

Curry Notation:

$$U_1 \rightarrow (U_2 \rightarrow (U_4 \rightarrow O_1 \,\&\, \langle U_1 \rangle) \,\&\, (U_5 \rightarrow \langle U_3 \rangle \,\&\, O_3) \,\&\, \langle U_3 \rangle) \,\&\, (U_3 \rightarrow O_2 \,\&\, O_1).$$

Polish notation:

$$\rightarrow_2 U_1 \rightarrow_3 U_2 \rightarrow_2 U_4\ O_1\ \langle U_1 \rangle \rightarrow_2 U_5\ \langle U_3 \rangle\ O_3\ \langle U_3 \rangle \rightarrow_2 U_3\ O_2\ \langle O_1 \rangle.$$

1949: "On the composition of programs for automatic computing"

Notations...

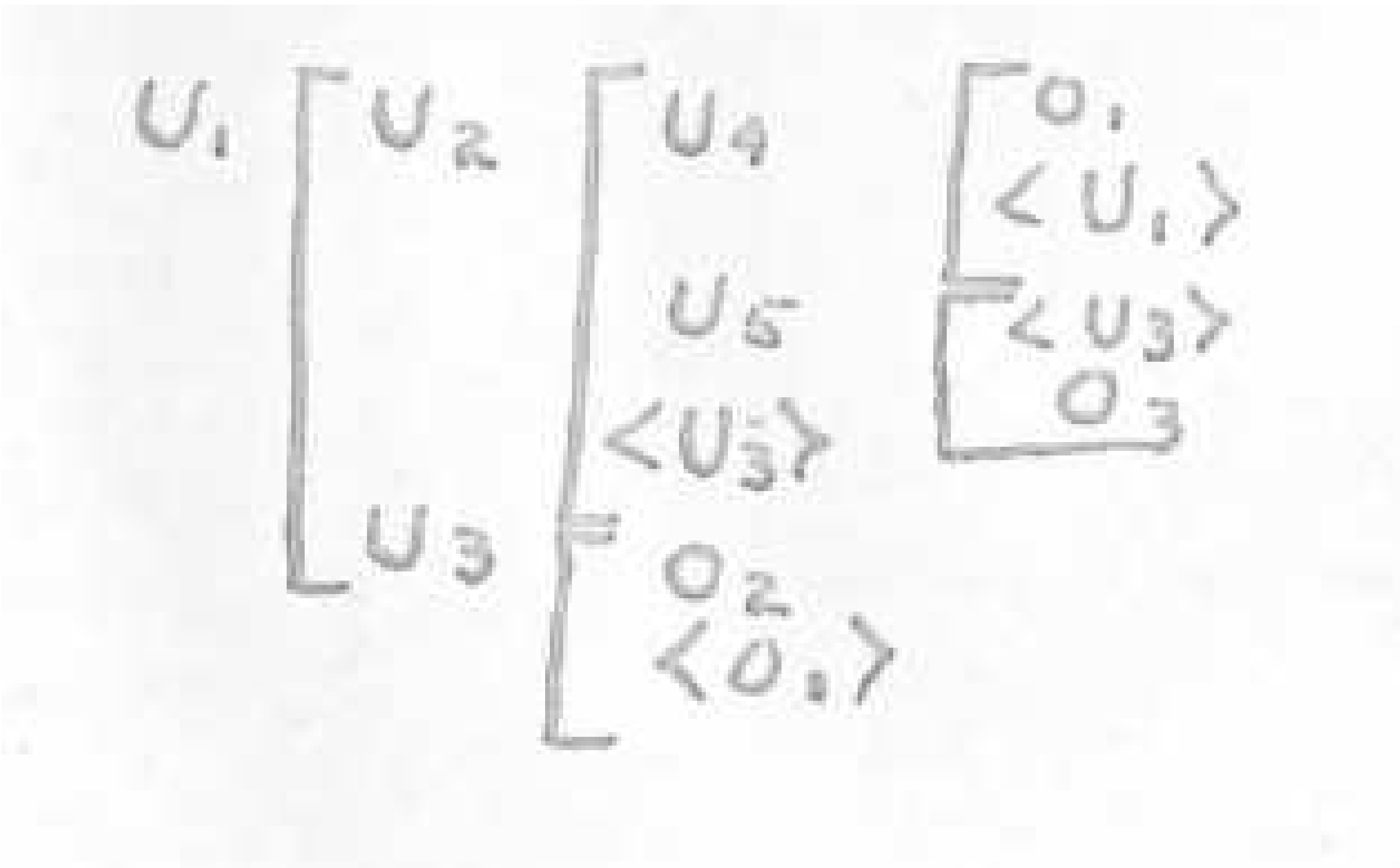
Peano notation:

$$U_1 \rightarrow : U_2 \rightarrow . U_4 \rightarrow O_1 \,\&\, \langle U_1 \rangle . \&. U_5 \rightarrow \langle U_3 \rangle$$
$$\&\, O_3 . \&. \langle U_3 \rangle : \& : U_3 \rightarrow O_2 \,\&\, \langle O_1 \rangle .$$

1949: "On the composition of programs for automatic computing"

Notations...

Begriffschrift:



"Frege's notation, it must be remembered, died with him"

1950: "A program composition technique as applied to inverse interpolation"
Some highlights

Synthesis of program of inverse interpolation: composition of the main routines of the problem

Analysis into **basic programs**: "This analysis can, in principle at least, be carried clear down until the ultimate constituents are the simplest possible programs [...] Of course, it is a platitude that the practical man would not be interested in composition techniques for programs of such simplicity, but i is a common experience in mathematics that one can deepen ones insight into the most profound and abstract theories by considering trivially simple examples."

Synthesis of basic programs (in general):

arithmetic programs: compiler for arithmetic procedures, i.e., "complete theory for the construction of an arbitrary such program. This program will not always be the shortest one possible to attain the required result; but, at least, it will be automatic as soon as certain decisions are made."

Discrimination programs

Secondary programs

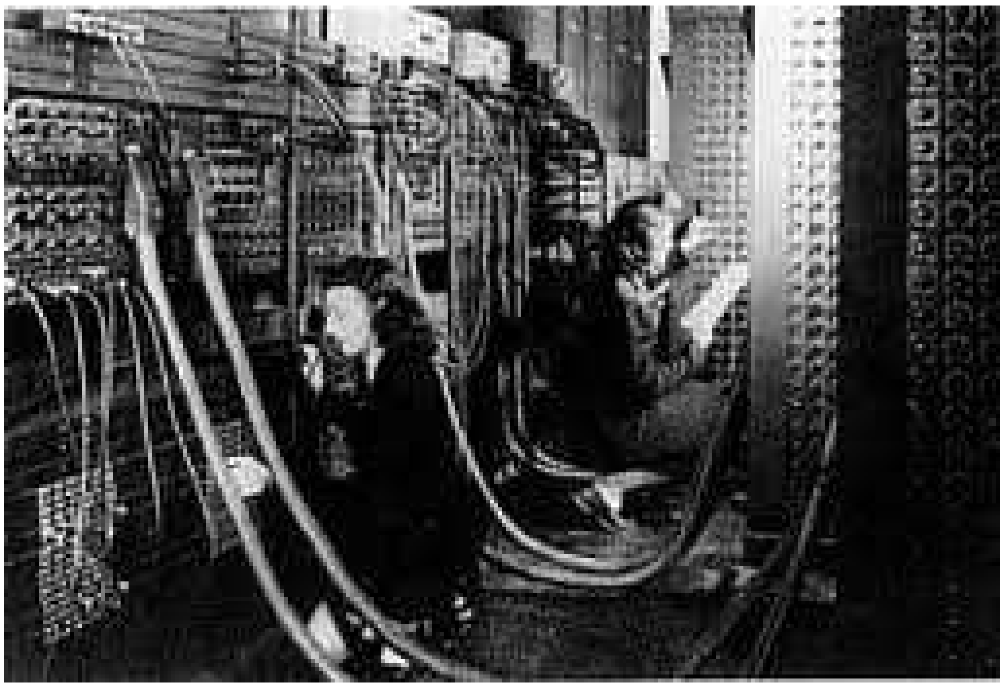
Table 1: Table of basic programs

Number for i =				**Symbol**	**effects**			**GvN for $i =$**			
0	**1**	**2**	**3**		A	R	X	**0**	**1**	**2**	**3**
1				$\{0 : A\}$	0	-	-	a			
2	3			$\{\pi_i(1) : A\}$	$\pi_i(1)$	-	-	a	a		
	4	5	6	$\{\pi_i(A) : A\}$	$\pi_i(A)$	-	-		a	a	a
7	8	9	10	$\pi_i(R) : A\}$	$\pi_i(R)$	R	-	A	a	a	a
11	12	13	14	$\{\pi_i(x) : A\}$	$\pi_i(x)$	-	x		-	M	-M
15				$\{d(*)\} : A\}$	$d(*)$		x	a			
16	17			$\{A + \pi_i(1) : A\}$	$A + \pi_i(1)$	-	-	a	a		
18	19	20	21	$\{A + \pi_i(R) : A\}$	$A + \pi_i(R)$	R	-	a	a	a	a
22	23	24	25	$\{A + \pi_i(x) : A\}$	$A + \pi_i(x)$	-	x	h	-h	Mh	-Mh
26				$\{A + d(*) : A\}$	$A + d(*)$	-	x	a			
27				$\{r\}$	$r(A)$		-	R			
28				$\{l\}$	$l(A)$		-	L			
29				$\{xR : A\}$	[a]	[a]	x	X			

Continued on next page

Table 1 – continued from previous page												
Number for i =				**Symbol**	**effects**			**GvN for** $i =$				
0	**1**	**2**	**3**		A	R	X	**0**	**1**	**2**	**3**
30				$\{A : R\}$	A	A	-	a			
31				$\{x : R\}$	A	-	A	R			
32				$\{A/x : R\}$	A	A	x	÷			
33				$\{A : x\}$	A	-	A	R			
34				$\{A : d(*)\}$	A	-	[b]	Sp			
35				$\{A : e(*)\}$	A	-	[b]	a			
36				$\{K\}$	-	-	-	C			
37				$\{A < 0\}$	-	-	-	Cc			
38				stop	-	-	-	a			

The ENIAC experience



The "ENIAC" experience: A new order of thinking?

- Speed + parallelism
- Internal "if" $\rightarrow$ *external* programming
- digital machine

$\Rightarrow$ Possibility of internalization & increased automation (from both sides)

$\Rightarrow$ Introduce discrete math in continuous math ($\sim$ Hartree)

$\Rightarrow$ "multi-purposeness"

$\Rightarrow$ "Interaction" through direct "sensory" contact, during set-up and calculation

$\Rightarrow$ laborious process of programming

The “ENIAC” experience for Lehmer, von Neumann and Curry

- **(explicit) Machine-awareness & centrism**
  - Lehmer: the “idiot” approach; “language” as a barrier; exploration of parallelism
  - Von Neumann: composition of programs largely manual; no “logical” theory of basic orders ;“the problem of coding routines need not and should not be a dominant difficulty”
  - Curry: no use of combinators, but new theory adapted to IAS machine; restrictions and assumptions; adaptation basic programs to limited memory; relativity of notation/language
- **Human-awareness & centrism**
  - Lehmer: Use of the machine for humanly impractical problems
  - Von Neumann: four stages of programming: only the first – mathematical preparation – is considered “difficult”; human practicality: “probably not want to produce large amounts of numerical material”
  - Curry: “[G]iven a certain memory capacity the principal bottleneck for efficient performance is the preparation of problems”

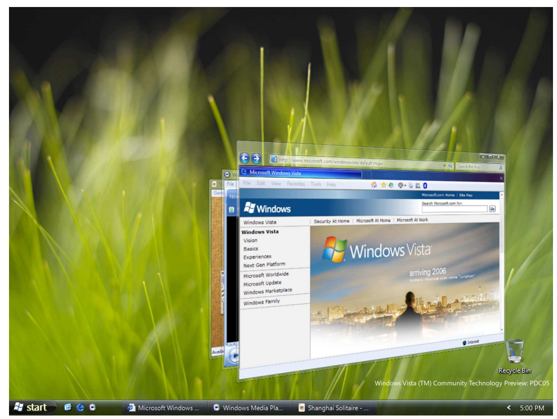
The "ENIAC" experience for Lehmer, von Neumann and Curry

- **Thinking within and beyond the man-machine limits**
  - Lehmer: slow process of punch input + limited memory: internal sieve as a heuristic program; computation composites done by hand; possibility of error; "This is not the kind of machine proof with a "look, no hands!" point of view [...] Rather it is a man-machine cooperative endeavor"
  - Von Neumann: problem of rounding-off+problem of error; logical representation of stored-program idea (Eckert, Mauchly); developing a plan for the computation; statistical analyses done by hand; theory of artificial and natural automata
  - Curry: "It is said that during the war an error in one of the firing tables was caused by using the wrong lead screw in the differential analyser. Such an error would have been impossible if the calculation had been completely programmed." "[C]onsequently features of machine design which will cause an improvement in programming technique should be very seriously considered "

The "ENIAC" experience for Lehmer, von Neumann and Curry

- **Automation and internalization**
  - Lehmer: sieve; possibility of the machine to do its own inspections
  - Von Neumann: pseudo-random generators (but!); "No complicated calculation can be carried out without storing considerable numerical material while the calculation is in progress"
  - Curry: "Now it is an important fact that the actual construction of a program indicated in the above symbolism is a mechanical process."

Confrontations with the modern computer



The computer – now

- exponential increase in speed and memory
- stored-program computers
- ease of "programming" ("user-friendly")
- graphical devices (printer, display)
- wider availbaility

$\Rightarrow$ high automatization and internalization

$\Rightarrow$ Increased interactivity

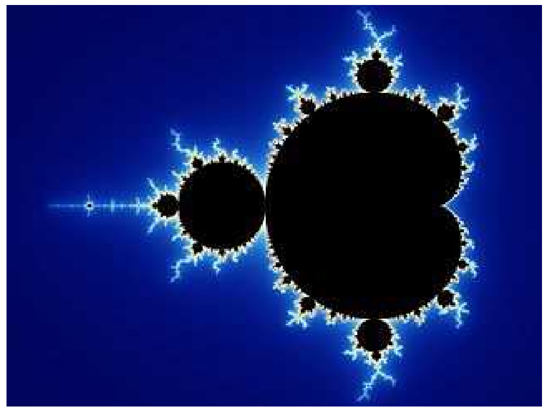
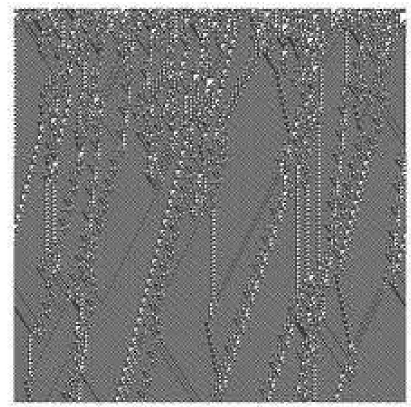
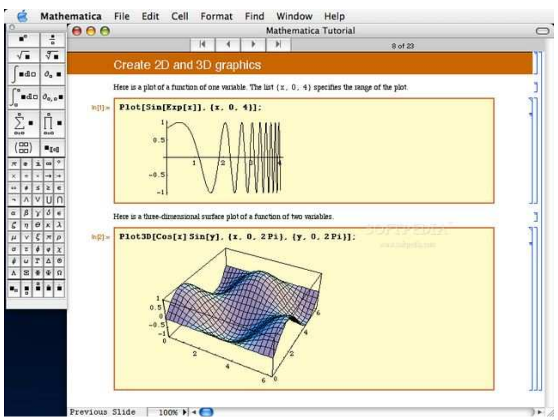
$\Rightarrow$ Indirect communication; restricted by the language; no "true" machine hacking

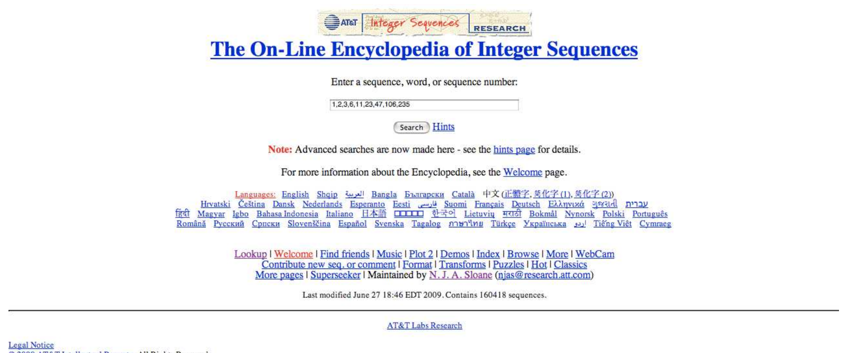
$\Rightarrow$ Increased responsability for the machine

The ENIAC-examples now

- Research on the random character of π and e: Bailey and Crandell, 2001 as a "typical" example of "experimental math" (PSLQ and BBP)
- Research on programming and compiling: a well-established discipline
- The use of (visual) models and simulations is legio in all disciplines of science (even philosophy!)
- The search for primes is ongoing: distributed computing

Three modern examples



Three modern examples

The visual: Mandelbrot and his set

– "[R]einvent the role of the eye" in math: "I look, look, look and play with many pictures"

– The computer as a (hidden) microscope: "Incidentally, a picture is like a reading of a scientific instrument"

– Emphasis on the insignificance of the quality of the printer/pictures: "specks of dust"' or interesting results?

Software: Wolfram, Mathematica & "A new kind of science"

– "[T]he visionary concept of Mathematica was to create once and for all a single system that could handle all the various aspects of technical computing–and beyond–in a coherent and unified way."

– "Maple and Mathematica have opened the door for an integrated process of experimentation, concept formation, and conjecturing"

– The development of an "integrated" science of everything, based on (visual and numerical) explorations of cellular automata

The computer as a database: Sloane and the encyclopedia of integer sequences

– An internet-based encyclopedia, with a built-in superseeker algorithm

– Human**s**-machine interactions: contributions by computers and humans

– Multifunctionality: to compute, to look-up, to solve, to educate, etc

Computer-assisted math now

- Machine-centrism? Rather "software"-centrism or the machine as a hidden (but more "responsable") instrument
- Human-centrism? User-friendliness in math (GUI's); "humanization" of math through experimental math;
- Thinking within the human-machine limits? Wolfram's principle of computational equivalence; significance of integrated and increased interactivity; risk of forgetting about the machine (taking it for granted): e.g. focus on the eye (mistakes in Wolfram's NKS);

Discussion

- The ENIAC-experience: the machine that triggered a new order of thinking within and because of the limits and possibilities _|¯
- “hands-on” vs. “hands-off”
  - Different kinds of mathematical thinking? Machine-centrism vs. human and software-centrism
  - Predefined knowledge in “hands-off’: more integrated but less necessary to know what is behind the name of an algorithm $\sim$ Husserl’s paradox of the progress of science
  - The sky is the limit in the hands-off approach? Machine-limits vs. theoretical and algorithmic limits; significance of being aware of (and being confronted with) the limitations

$\Rightarrow$ ”The lesson seems to be this: we cannot fully understand our own conceptual scheme withouth plumbing its historical roots”