A Very High-Speed Digital Number Sieve

By D. G. Cantor, G. Estrin, A. S. Fraenkel, and R. Turn

1. Introduction. The general sieve problem may be stated as follows [3]. Let $m_1, m_2, \cdots, m_s$ be s positive integers, relatively prime in pairs. Consider the congruences

$$x \equiv a_{ij} (\mathrm{mod}\ m_i), \qquad i = 1, 2, \cdots, s; j = 1, 2, \cdots, t_i < m_i. \tag{1}$$

For fixed i, the a_{ij} are distinct non-negative integers less than m_i. The problem is to find all integers N between given limits, say

$$A \leqq N < B, \tag{2}$$

such that N is a solution to s of the congruences. (It is, of course, clear that no N can be a solution to more than s of the congruences (1).)

Examples: On the one extreme there is the *Sieve of Eratosthenes* for finding all primes p in the range $A = B^{1/2} \leqq p < B$, where $t_i = m_i - 1$ for all i. (Here m_i are all the primes $< B^{1/2}$.) On the other extreme there is the Chinese remainder type of problem, where $t_i = 1$ for all i, and there is only one solution among $\prod_{i=1}^{s} m_i$ numbers.

In between these two extremes, there is the important *quadratic sieve*, where roughly $t_i = m_i/2$ for all i. It is used in problems involving quadratic residues, Diophantine equations of second degree and other quadratic type problems.

About thirty years ago, Lehmer [1], [2] constructed a novel special-purpose device for sifting. It used the first 30 primes as moduli. Its processing rate was

$$3 \times 10^5 \text{ numbers/min.}$$

General-purpose computers are not very well suited to sifting, and the earlier models could not compete with Lehmer's machine. However, the speed of the more recent machines makes up for their lack of orientation towards the sieve problem insofar as surpassing the performance of Lehmer's machine is concerned. Thus, the rate for a quadratic sieve using the first 30 primes on the IBM 7090 is approximately

$$10^7 \text{ numbers/min.}$$

The present paper describes a special-purpose device, where rates in excess of

$$10^{10} \text{ numbers/min.}$$

can be achieved for quadratic sieves. The device consists of basic digital building blocks from which a suitable sieve is assembled for each problem. Thus, by an appropriate rearrangement of the building blocks, problems with different moduli can be run. It is also shown that if the device contains a certain minimum amount of hardware and is attached to a general-purpose computer, then problems can be run where, roughly speaking, the number of moduli is not limited any more by the

Received June 23, 1961. The preparation of this paper was sponsored, in part, by the Office of Naval Research and the Atomic Energy Commission.

amount of hardware of the special-purpose device, but only by the size of the memory of the general-purpose computer, and the rate is still of the order of 10^{10} numbers/min. We use a so-called "Fixed Plus Variable Structure Computer" organization for realizing the combination between the special- and general-purpose computers [6]. This also enables one to use the digital building blocks of the sieve for building other special-purpose devices which one might want to associate with the general-purpose computer.

2. Binary Set-Up of the Sieve. For solving the system (1) on a digital computer, we consider a matrix M of size $s \times (B - A)$ with entries c_{ik} $(i = 1, 2, \cdots, s; k = A, A + 1, \cdots, B - 1)$, defined by

$$c_{ik} = \begin{cases} 1 \text{ if } k \equiv a_{ij} \pmod{m_i} \\ 0 \text{ otherwise} \end{cases} \qquad (j = 1, 2, \cdots, t_i).$$

Then every column, all of whose entries are 1, corresponds to a solution, and conversely.

Example: Find the primes p such that

$$6 \leqq p < 36. \tag{3}$$

The relevant congruences are $x \equiv 1 \pmod 2$, $x \equiv 1, 2 \pmod 3$, $x \equiv 1, 2, 3, 4 \pmod 5$. The matrix M is given by

m_i \ N	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
2	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
3	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1	0	1	1
5	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0	1	1	1	1	0

The columns all of whose entries are 1 correspond to the primes in the range (3).

The rows of M are periodic with period m_i. Thus, the first ordered m_i bits of the ith row determine the rest of this row completely, and we call them the *periodic pattern* e_i of m_i.

3. Method of Solution on the Special-Purpose Computer. We now give an informal introduction to the principle of operation of the special-purpose device. It will be observed that the method is based on ideas used in earlier work [1], [2], [3] in this field.

A first approach to the mechanization of a special-purpose sieve would be to build a matrix precisely in the form displayed for the example above with observation posts in every column, detecting coincidences of non-zero bits. Problems which can be solved by such a procedure are limited to those which can fit into the maximum size matrix which can be assembled, i.e., computationally trivial problems.

We order the moduli so that

$$m_1 < m_2 < \cdots < m_s. \tag{4}$$

Since solutions exist only corresponding to columns with non-zero bits, we may eliminate the m_s-row and many of the components required to detect coincidences, by establishing coincidence gates only in those columns where the m_s-row has non-zero entries.

Large problems may be handled by constructing only m_s columns of the matrix and testing for solutions in these columns in parallel. Entering the next batch of m_s numbers turns out to be equivalent to performing prearranged circular shifts in the $s - 1$ rows. This procedure would require a matrix of size $(s - 1) \times m_s$ with coincidence gates established in the columns as prescribed above. It is possible to use such a matrix to define potential solutions even when it is only feasible to mechanize $l < s - 1$ rows of the matrix, and then have a general-purpose computer complete the test for solution.

The range of problems which may be handled is increased when it is recognized that the periodic pattern e_i associated with the ith row completely determines the rest of the row. In the following we give an algorithm which defines a procedure requiring only m_i elements in the ith row, giving up only the regularity of the coincident gate connections. The special-purpose computer consists of basic digital building blocks or modules which are assembled into a matrix consisting of $s - 1$ shifting registers, the ith of length m_i, and initially containing the periodic pattern e_i $(i = 1, 2, \cdots, s - 1)$. Observation posts are placed at certain positions in the matrix which sift out the solutions to (1) among the first m_s numbers.* Next, a circular shift is performed in each register, which is equivalent to bringing in the next m_s numbers to be sifted. This is followed by the observation posts sifting out the solutions among this new batch of numbers. This process of sifting followed by shifting is continued until all the numbers are processed.

4. The Algorithm. We divide the numbers N in the range (2) into sets S_n defined by†

$$(5)\quad S_n = \{N : N < B, N = A + nm_s + k; 0 \leqq k < m_s\},\quad n = 0, 1, \cdots, \left[\frac{B - A - 1}{m_s}\right].$$

Thus, each set (except possibly the last) contains m_s numbers.

Let

$$(6)\qquad m_s = q_i m_i + r_i, \qquad 0 < r_i < m_i \quad (i = 1, 2, \cdots, s - 1).$$

With each set S_n we associate a matrix M_n of size $(s - 1) \times m_s$ with entries $c_{ij}(n)$ $(i = 1, 2, \cdots, s - 1; j = 0, 1, \cdots, m_s - 1)$, defined recursively by

$$(7)\quad c_{ij}(0) = \begin{cases} 1 \text{ if } 0 \leqq j < m_i \text{ and } A + j \equiv a_{ij}, \cdots, a_{i,\,ti} \pmod{m_i} \\ 0 \text{ otherwise.} \end{cases}$$

$$(8)\qquad c_{ij}(n) = \begin{cases} c_{i,\,j+r_i}(n - 1) & \text{if } 0 \leqq j + r_i < m_i \\ c_{i,\,j+r_i-m_i}(n - 1) & \text{if } 0 \leqq j < m_i \text{ and } j + r_i \geqq m_i \\ 0 & \text{if } m_i \leqq j < m_s. \end{cases}$$

* The positioning of the observation posts is determined by m_s and its residues in such a way that a register of length m_s is not required.

† $[x]$ stands for the largest integer $\leqq x$.

Equation (8) can be written in the form

$$c_{ij}(n) = \begin{cases} c_{id}(n-1) & \text{where } m_i > d \equiv j + r_i \pmod{m_i}, \text{ if } 0 \leqq j < m_i \\ 0 & \text{if } m_i \leqq j < m_s . \end{cases}$$

Hence (7) and (8) are equivalent to

$$(9) \quad c_{ij}(n) = \begin{cases} 1 & \text{if } 0 \leqq j < m_i \text{ and } A + j \equiv a_{i,v_i} - nr_i \pmod{m_i} \\ 0 & \text{otherwise} \end{cases} \qquad (v_i = 1, 2, \cdots, t_i).$$

If $N \in S_n$ is a solution to the system (1), then by (5),

$$(10) \qquad A + k \equiv a_{s,v_s} \pmod{m_s} \quad (0 \leqq k < m_s ; v_s = 1, 2, \cdots, t_s)$$

for all n.
By (5) and (6) we have also

$$(11) \quad A + k \equiv a_{i,v_i} - nr_i \pmod{m_i} \quad (v_i = 1, 2, \cdots, t_i ; i = 1, 2, \cdots, s-1).$$

Hence, if we let

$$(12) \qquad k = w_i m_i + u_i , \qquad 0 \leqq u_i < m_i \qquad (i = 1, 2, \cdots, s-1),$$

then by (11),

$$A + u_i \equiv a_{i,v_i} - nr_i \pmod{m_i} \qquad (v_i = 1, 2, \cdots, t_i ; i = 1, 2, \cdots, s-1),$$

so that

$$(13) \qquad c_{i,u_i}(n) = 1$$

for $i = 1, \cdots, s-1$ by (9).

Also the converse holds. That is to say, if (13) holds for $i = 1, \cdots, s-1$ (where u_i is given by (12) and k by (10)), then

$$(14) \qquad N = A + nm_s + k$$

is a solution to the system (1). This is the basis of the algorithm. We list the t_s solutions $k_1, \cdots, k_{t_s}$ of (10), and for each of them its corresponding $s-1$ values u_i. Then the numbers $N = A + nm_s + k_{v_s}$ $(v_s = 1, \cdots, t_s)$ for which (13) holds for $i = 1, \cdots, s-1$, are solutions to (1), and these are all the solutions.

Example: Find all solutions in the range

$$-15 \leqq N < 25$$

to the system of congruences

$$\begin{array}{ll} x \equiv 1 & \pmod 2 \\ x \equiv 1, 2 & \pmod 3 \\ x \equiv 2, 3, 4 & \pmod 5 \\ x \equiv 0, 1, 2, 5 & \pmod 7 \\ x \equiv 0, 1, 8, 9 & \pmod{11}. \end{array}$$

Reference is made to Table I. The matrix M_0 is constructed according to (7) (omitting all strings of zeros). Equation (8) implies that the ith row of matrix M_n is obtained from the ith row of M_{n-1} by means of circularly left shifting the

TABLE I

The Matrices for the Problem

i \ N	−15	−14	−13	−12	−11	−10	−9	−8	−7	−6	−5
	0	1	2	3	4	5	6	7	8	9	10
1	**1**	0									
2	0	1	**1**						M_0		
3	0	0	**1**	1	1						
4	0	1	**1**	1	0	0	1				

i \ N	−4	−3	−2	−1	0	1	2	3	4	5	6
	0	1	2	3	4	5	6	7	8	9	10
1	0	1									
2	1	0	1						M_1		
3	0	1	1	1	0						
4	0	0	1	0	1	1	1				

i \ N	7	8	9	10	11	12	13	14	15	16	17
	0	1	2	3	4	5	6	7	8	9	10
1	1	0									
2	1	1	0						M_2		
3	1	1	1	0	0						
4	1	1	1	0	0	1	0				

i \ N	18	19	20	21	22	23	24
	0	1	2	3	4	5	6
1	0	**1**					
2	0	**1**	**1**			M_3	
3	**1**	**1**	0	0	1		
4	0	**1**	0	1	1	**1**	0

first m_i bits by r_i positions (or by circularly right shifting them by $m_i - r_i$ positions). In the present case, $r_1 = 1$, $r_2 = 2$, $r_3 = 1$, $r_4 = 4$, so that in passing from one matrix to the next, the first row is shifted left circularly by 1 position, the second by 2, the third by 1 and the fourth by 4 positions. This is the way M_1, M_2 and M_3 are obtained.

The values k_1, k_2, k_3, k_4 of Table II are computed by (10), and the corresponding values of u_i by (12). Table II defines a pattern of observation stations which sift out the solutions in each matrix; the entry u_i represents the column coordinate corresponding to the row coordinate i, at which an observation station exists. In M_0, the observation pattern $u_i = 0, 2, 2, 2$ indicates a solution, since all matrix positions corresponding to that pattern are filled by 1's. They appear in bold type in Table I. The corresponding value of k is $k_4 = 2$. The solution is, therefore, $N = -15 + 0 \times 11 + 2 = -13$ by (14). The only two other solutions

TABLE II
Observation Posts for the Problem

i	m_i	U_i			
		$k_1 = 4$	$k_2 = 5$	$k_3 = 1$	$k_4 = 2$
1	2	0	1	1	0
2	3	1	2	1	2
3	5	4	0	1	2
4	7	4	5	1	2

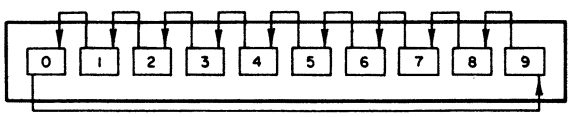

FIG. 1.—Conventional Shift Register.

are found in M_3, for $k_3 = 1$ and $k_2 = 5$, also indicated by bold type in Table I. They are $N = -15 + 3 \times 11 + 1 = 19$ and $N = -15 + 3 \times 11 + 5 = 23$.

5. The Special-Purpose Computer. The special-purpose device should be flexible enough to allow usage of different moduli. Therefore the basic building blocks of the device are modules consisting of memory elements and gates which will be assembled into the appropriate-size shifting registers and sets of and-gates for each problem. The and-gates are the realization of appropriate combinations of the observation posts, and test for coincidence of 1's.

The example of the previous section suggests the construction of the device. Its central part consists of shift-registers $R_1, \cdots, R_{s-1}$, the ith of length m_i, which will store the matrices M_n (without the trailing strings of 0's). Register R_i will shift circularly left by r_i positions. It is important to note, however, that this can be effected in one shift time, rather than in r_i shift times, and further, that the wiring can be so arranged that any transmitting memory element is adjacent to its receiver. In order to do this, we rename the memory elements in R_i so that element number 0 is at the left, followed by element number r_i, followed by number $2r_i \pmod{m_i}$, by $3r_i \pmod{m_i}$, etc. Then each transmitting element is adjacent to its receiver, and every element will appear exactly once. For‡ $(m_i, m_s) = 1$, so that also $(m_i, r_i) = 1$ $(i = 1, \cdots, s - 1)$. Hence r_i generates the additive cyclic group of non-negative integers (mod m_i) and every element appears exactly once in the register.

Example: Suppose that for a certain sieve problem $m_s = 23$, and $m_i = 10$ for some $i < s$. Then R_i has to shift circularly left by $r_i = 3$ positions. On a conventional shift-register, three shifts of the type indicated in Figure 1 would have to be performed. The same result can be obtained in one shift time by the specially wired-up register of Figure 2. However, the long wiring has an undesirable effect on the speed of the system. Renaming the memory elements as in Figure 3, the three shifts can be done in one shift time with conventional wiring.

‡ (a, b) stands for the greatest common divisor of a and b.

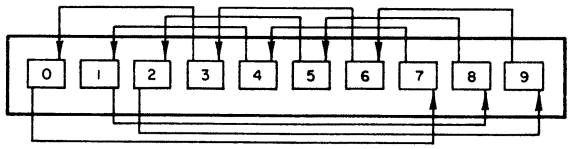

Fig. 2.—Specially Wired-Up Register for Performing a Shift of Three Positions in One Shift Time.

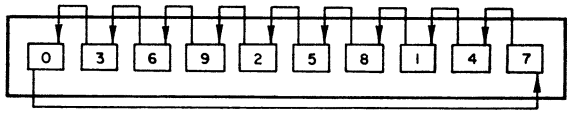

FIG. 3.—Final Form of the Register.

The second part of the special-purpose device consists of t_s sets of and-gates, one set for each value of k which is a solution to (10), "anding" together positions c_{ij} in the registers, as defined in the previous section. Also a counting register R with capacity $> (B - A)/m_s$ and a shift control are required. (The shift control does not normally have to be reconstructed for every problem.) The register R contains the value n of the matrix M_n currently being sifted.

The sifting process consists of the following steps.

1. Clear R.
2. Load $R_1, \cdots, R_{s-1}$ with M_0, whose entries are defined by (7).
3. Record n and record the values k for which coincidence is obtained, i.e., the values k associated with the sets of and-gates which are excited, if any. The corresponding solutions are given by (14).
4. Advance R by unity and perform a circular shift in each shift-register according to the above scheme. This effectively reloads the registers with the next batch of m_s numbers to be sifted.
5. Terminate process if $n > (B - A)/m_s$. Otherwise go back to 3.

It is thus seen that in this process m_s numbers are processed per shift time.

6. The Sieve in a Fixed Plus Variable Structure Computer. It was remarked by Lehmer that special-purpose equipment attached to the arithmetic unit of a fast computer can speed up computation of permutation problems [4], and of other problems [5]. More generally, we consider a so-called "Fixed Plus Variable Structure Computer" (to be designated by $(F + V)$ computer), which consists of a conventional digital computer (the fixed part to be denoted by F), and a set of modules (the variable part to be denoted by V). Many problems contain a part which can be solved on a special-purpose computer in a much more efficient way than on a general-purpose computer. For such a problem, the modules are assembled into a suitable special-purpose device which handles this part. The rest of the problem is handled by F. A supervisory control coordinates the operation of the two computers. However, the special-purpose configuration is not retained permanently, but may be reorganized into other configurations for other problems. For a more detailed description of the concept of the $(F + V)$ computer, the reader is referred to the literature [6].

The special-purpose device described above has a limited amount of hardware.

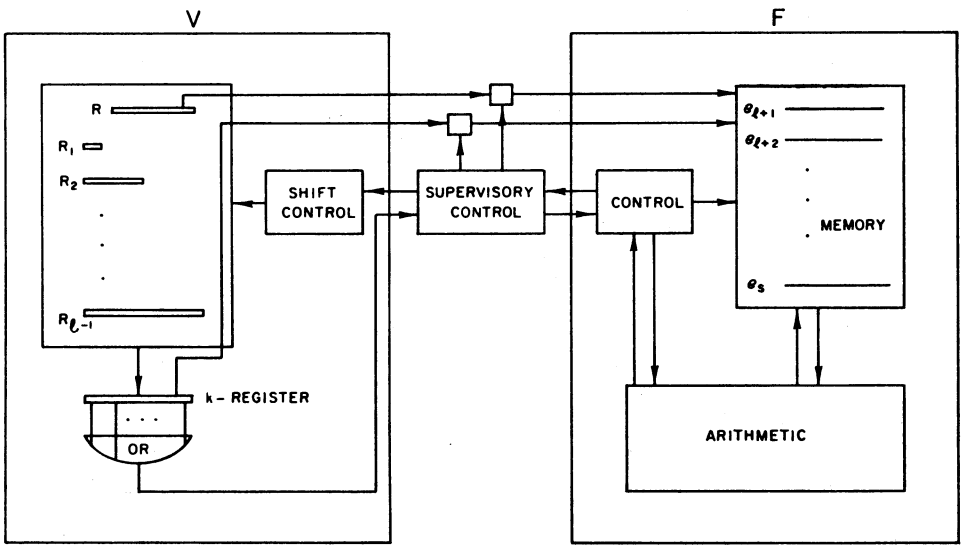

FIG. 4.—($F + V$) Computer Organization for the Sieve Problem.

For certain problems it may be desirable to use more moduli than can be mechanized with the available hardware. In order to handle such problems, we imbed the special-purpose device in an ($F + V$) computer. This allows, as will be shown subsequently, handling problems in which the number of moduli is limited only by the number of periodic patterns e_i of m_i that can be stored in the memory of F, provided that the hardware of V is sufficient to mechanize the first l of the s moduli. The parameter l depends on the relative number of 1's and 0's in the periodic patterns of the sieve, and on the relative speeds of F and V. The use of $F + V$ also allows using the modules for building special-purpose devices other than the sieve, and attaching them to F.

Figure 4 shows the organization of the ($F + V$) computer for the sieve problem by means of a block diagram. The V-part, which acts as the special-purpose device, mechanizes the first l moduli. The k-register records the values of k (the solutions of (10)) corresponding to coincidence. The periodic patterns $e_{l+1}, \cdots, e_s$ of $m_{l+1}, \cdots, m_s$ are stored in the memory of F. Numbers will be sifted in V, and when coincidence occurs, the contents of R and of the k-register are transferred to F, where so-called *solution candidates* N of the form (14) are formed, one for each value of k. Then divisions of the form

$$N = \sigma_i m_i + \rho_i, \qquad 0 \leqq \rho_i < m_i \qquad (i = l + 1, \cdots, s) \tag{15}$$

are performed in F. The residue ρ_i determines uniquely the position of the bit of the periodic pattern e_i of m_i corresponding to N. The number N is a solution if and only if these bits are 1 for $i = l + 1, \cdots, s$. Thus, the ($F + V$) computer is so organized that V will do the high-speed sifting, and F will do divisions. l will be chosen so that the average time per coincidence in V is at least as large as the average division time per coincidence in F. Then V would normally do its divisions, until such time when coincidence is obtained in V. At such time, V is interrupted

and the transfers from V to the memory of F occur, the latter acting as a buffer, capable of storing "bursts" of solution candidates which V might produce occasionally. After the transfers, both parts are again decoupled and assume their respective tasks. The program for V is outlined in the following six steps.

1. Clear R.
2. Load $R_1, \cdots, R_{l-1}$ with M_0.
3. Check for coincidence. If none is obtained, go to 5. Otherwise continue with 4.
4. Interrupt F and V. Transfer the contents of R and of the k-registers to the memory of F.
5. Advance R by unity and perform a circular left shift of r_i places in R_i ($i = 1, \cdots, l-1$).
6. Terminate process if $n > (B - A)/m_l$. Otherwise go back to 3.

The program for F is simply to produce solution candidates N of the form (14), and to perform divisions of the type (15) for each of them until the first 0-bit is encountered. If none is encountered, N is recorded as a solution.

7. Speed and Hardware. The speed and hardware requirements will now be discussed in terms of an example, for which we choose a quadratic sieve problem where the moduli are the first s primes. The first column of Table III contains values of the independent variable l, the number of moduli mechanized in the special-purpose device. The table displays the speed and hardware requirements for such a sieve as a function of l. The second column contains the lth prime. Let t be the total time required for performing the coincidence test and the subsequent circular shifts in the registers. Using the register organization described in Section 5, the circular shifting amounts to a left shift of one position in each register. We assume transistorized circuitry, for which

$$t = 0.2\ \mu \text{ sec}$$

is chosen (speeds approximating those of the IBM 7090). Thus, m_l numbers are processed in this time if no coincidence occurs. (If the registers are of the double-rank type, both ranks will be equipped with sets of and-gates, and $t = 0.2\ \mu$ sec is the time for a coincidence check and for transferring one rank into the other. Thus also in this case m_l numbers are processed in $0.2\ \mu$ sec.)

We consider first the case $s = l$, that is, we use only a special-purpose computer without a conventional general-purpose computer. Assuming the solutions to be sparse, so that we may neglect the time of recording them, the rate of the sieve is

$$v = \frac{6 \times 10^7 \times m_l}{t} = 3 \times 10^8 \times m_l \text{ numbers/min.}$$

These values are displayed in the third column.

Since the sieve is quadratic, the probability of any randomly selected bit in the periodic pattern e_i to be 1 is about 0.5. Hence, on the average, one coincidence is obtained per 2^l numbers sifted, or every

$$\tau = \frac{6 \times 10^7 \times 2^l}{v} = \frac{2^l}{5m_l}\ \mu \text{ sec.}$$

These values appear in the fourth column.

If $s > l$, divisions have to be performed in F. The probability that exactly i divisions suffice to decide whether any solution candidate N has to be rejected or accepted is $(\frac{1}{2})^i$ $(1 \leqq i \leqq s - l)$. Hence the expectation of the number of divisions for each N is given by

$$v = \sum_{i=1}^{s-l} i/2^i = 2 - (s - l + 2)/2^{s-l}. \tag{16}$$

Thus the average number of divisions for each solution candidate approaches 2 asymptotically from below. Assuming the IBM 7090 as the fixed machine F, this division subroutine takes about 200 μ sec for two divisions. Also, preliminary studies of the mode of transfer from V to the 7090 indicate that the transfer of the contents of R and of the k-register requires no more than 7 μ sec. (See appendix.) That is, this is the maximum time during which V is idle. F is interrupted only insofar as it requires memory access during this time. Actually, V could already resume its operation after the transfer of n from the R-register. It would have to wait additional time only if a new solution candidate is formed before the current contents of the k-register has been stored away, which is a rare event. However, in our computation of the overall speed of the sieve we assumed that V is interrupted for 7 μ sec. during each transfer.

Thus, the average overall rate of the sieve is given by

$$w = \frac{v}{1 + 7/\tau} \text{ numbers/min.}$$

for $\tau \geqq 200$ μ sec. If $\tau < 200$ μ sec, V will have to wait for F, and the average overall rate for this case is

$$w = \frac{\tau}{200} \frac{v}{1 + 7/\tau} \text{ numbers/min.}$$

Thus, the operation of the sieve becomes rapidly more and more inefficient as τ decreases below the critical value of 200 μ sec. The values of w appear in the fifth column of Table III. The last column displays the required number $h = \sum_{i=1}^{l-1} m_i$ of memory elements for the special-purpose device. (This number has to be doubled if the registers are of the double-rank type.)

The lower bound for l in Table III was chosen to be 9 because for $l = 8$ the rate would already be less than can be achieved with conventional present-day computers. The upper bound was chosen by setting arbitrarily a hardware constraint of 1500 memory elements.

The lowest value of τ for which $\tau \geqq 200$ μ sec is $\tau = 247.3$ μ sec. Thus V should contain at least 15 registers consisting of 328 memory elements. Figure 5 displays the overall rate as a function of required memory elements. Two simple conclusions can be drawn from the monotonicity of w as displayed in Figure 5. First, l should be chosen as large as possible. That is to say, as much hardware as available should be thrown in to build the sieve; even so the cooperation between F and V becomes less efficient as l increases beyond the critical value of 16, in the sense that F becomes more idle. Secondly, the use of slower memory elements is indicated if a larger number of them is available, hence the possibility of using magnetic core registers.

TABLE III

Speed and Hardware as a Function of l For a Quadratic Sieve

l—No. of Moduli Implemented in Special-Purpose Device	m_l—the lth Modulus	v—Numbers/ Min. Rate of Sieve if $s = l$	τ Average Time Per Coincidence	w—Numbers/ Min. Rate of Sieve if $s > l$	h—Number of Memory Elements in Special-Purpose Device
9	23	6.9×10^9	4.5 μ sec	5.70×10^7	77
10	29	8.7×10^9	7.1 μ sec	1.56×10^8	100
11	31	9.3×10^9	13.2 μ sec	4.03×10^8	129
12	37	1.11×10^{10}	22.1 μ sec	9.28×10^8	160
13	41	1.23×10^{10}	39.9 μ sec	2.09×10^9	197
14	43	1.29×10^{10}	76.2 μ sec	4.49×10^9	238
15	47	1.41×10^{10}	139.4 μ sec	9.34×10^9	281
16	53	1.59×10^{10}	247.3 μ sec	1.54×10^{10}	328
17	59	1.77×10^{10}	444.3 μ sec	1.73×10^{10}	381
18	61	1.83×10^{10}	859 μ sec	1.81×10^{10}	440
19	67	2.01×10^{10}	1.6 m sec	2.01×10^{10}	501
20	71	2.13×10^{10}	2.9 m sec	2.13×10^{10}	568
21	73	2.19×10^{10}	5.7 m sec	2.19×10^{10}	639
22	79	2.37×10^{10}	10.6 m sec	2.37×10^{10}	712
23	83	2.49×10^{10}	20.2 m sec	2.49×10^{10}	791
24	89	2.67×10^{10}	37.7 m sec	2.67×10^{10}	874
25	97	2.91×10^{10}	69.2 m sec	2.91×10^{10}	963
26	101	3.03×10^{10}	132.9 m sec	3.03×10^{10}	1060
27	103	3.09×10^{10}	260.6 m sec	3.09×10^{10}	1161
28	107	3.21×10^{10}	501.7 m sec	3.21×10^{10}	1264
29	109	3.27×10^{10}	985.1 m sec	3.27×10^{10}	1371
30	113	3.39×10^{10}	1900 m sec	3.39×10^{10}	1480

By (16), the average number of divisions per solution candidate in F is less than 2, whatever the number of periodic patterns that are stored in F. Therefore, the rate w of a quadratic sieve is practically independent of s, and the number of the moduli is limited only by the number of periods that can be stored in F. A similar remark applies also for sieves that are "less than quadratic," i.e., where the number of 1's in e_i is $< m_i/2$. For these types of sieves there are even less divisions to be performed, and a higher overall rate is obtained. For sieves that are "more than quadratic," and in particular for those which approach the type of sieve of Eratosthenes, more than two divisions are required on the average, and a higher critical value of l is obtained.

Variations of the above described method which result in even higher speeds (and therefore involve higher critical values of l) are clearly possible. For example, two moduli may be combined in V, say m_l and m_{l-1}, by initially solving the two congruences involving m_l and m_{l-1} manually or on F. Then $m_l m_{l-1}$ numbers can be processed per shift time, for which $t_l t_{l-1}$ sets of and-gates are required. Registers m_l and m_{l-1} do not have to be built of course. As another example, we might mechanize the moduli $m_1, m_2, \cdots, m_{l-1}, m_s$ in V, rather than $m_1, m_2, \cdots, m_{l-1}, m_l$, so that m_s numbers instead of only m_l are processed per shift time.

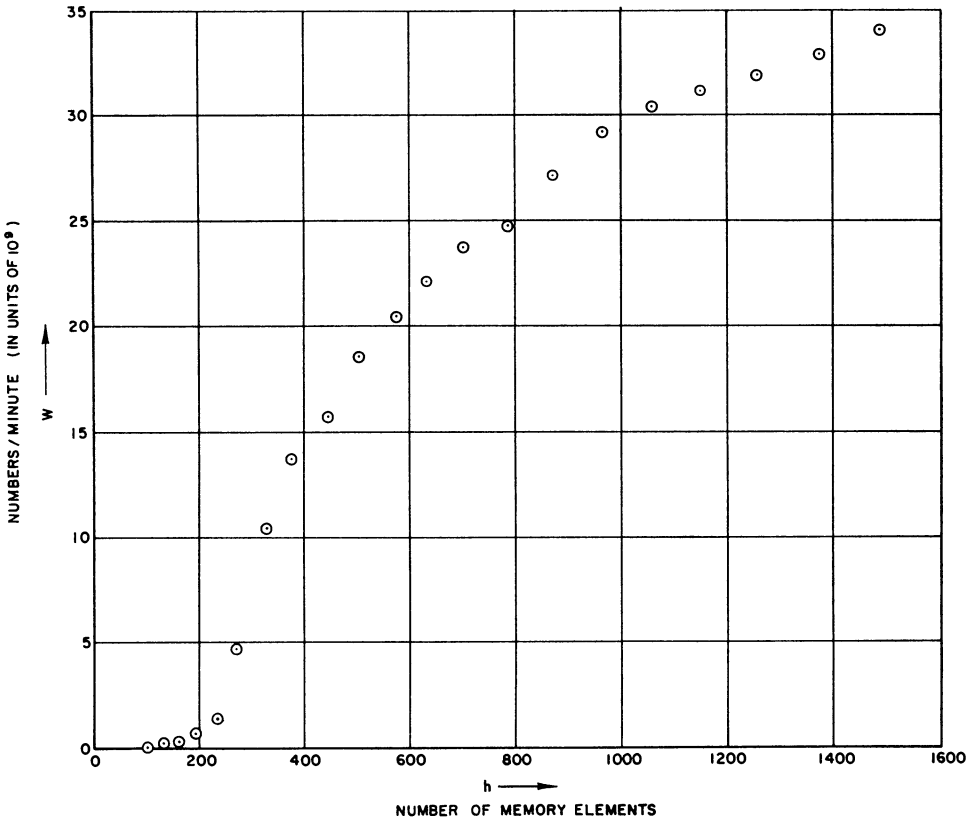

FIG. 5.—Rate of Sieve as a Function of the Number of Memory Elements.

Similarly, two moduli may be combined in F. For example, combining m_{l+1} with m_{l+2} and m_{l+3} with m_{l+4} (which increases storage requirements in F), and using as divisors the moduli $m_{l+1}m_{l+2}$, $m_{l+3}m_{l+4}$, m_{l+5}, $\cdots$, m_s, the average number of divisions that have to be performed approaches 11/8 asymptotically from below. Thus, the effect of combining moduli in F is to lower the critical value of τ. Such a procedure would therefore be used when the available hardware in V is smaller than required for keeping up with the speed of F implied by an average of two divisions per solution candidate.

The high speeds which can be achieved by our method suggest its applicability for conversion of numbers from the modular number representation [7] to the conventional polyadic representation. Since this problem is of the Chinese remainder type, it seems possible to include in the sieve special solution hunting properties.

8. Conclusion. A method has been presented to sift numbers satisfying a set of linear congruences from among a large set of numbers. The important properties of the resulting special-purpose device are that a relatively large set of numbers is processed essentially within the time required for performing a shift of one position in an ordinary shift-register, and that no memory references are necessary. This leads to an overall speed gain of about three orders of magnitude over modern

present-day computers such as the IBM 7090. By combining the device with a general-purpose computer, the size of problems that can be run is greatly increased with almost no decrease in speed.

Appendix

The Division Subroutine. For the purposes of this subroutine, written for the IBM 7090, we restrict the size of N in (2) to a number representable by 72 binary bits.

The first 36 of these are called HW, and the last 36 are called LW. The core of the subroutine consists of the following sequence, where it is assumed that the accumulator is cleared at the beginning. Every bit of the periodic patterns e_i is stored in a separate word, denoted by WM, and the corresponding period is stored in M.

LDQ	(Load the MQ)	HW
DVP	(Divide)	M
LDQ	(Load the MQ)	LW
DVP	(Divide)	M
PAC	(Place complement of address in index register)	0, 4
CLA	(Clear add)	WM, 4
TMI	(Transfer on minus)	OUT

This sequence requires 36 cycles. For a quadratic sieve, the sequence has to be performed twice on the average for each solution candidate. Another 20 cycles are required for performing the multiplication and addition implied by (14) and bookkeeping. One cycle takes 2.18 μ sec. Thus the subroutine requires about 200 μ sec.

Transfers from V to F. A preliminary study of the$(F + V)$ organization based on the IBM 7090 as F indicates that transfers from V to F can be effected in the manner of a data channel. Such a channel has a "Channel Address Counter" (CAC), from which addresses are transferred to the "Memory Address Register."

Suppose that the memory region bounded by addresses K and $K + M$ is allocated for storing the value n contained in R, and L to $L + M$ for storing the contents of the k-register. We assume, for simplicity, that registers R and k do not exceed 36 bits. In its normal form, the CAC contains an address of the form $K + i$ $(0 \leqq i \leqq M)$. Three flip-flops $FF1$, $FF2$, $FF3$ are contained in SC. $FF1$ records whether F or V was the last user of the buffer region of the memory. $FF2$ and $FF3$ define "full" and "empty" conditions of the buffer.

We adopt the following operating rules.

1. When V wants to store into the memory, the CAC is advanced by 1 if $FF1 = 1$, and remains unchanged if $FF1 = 0$. Then n is stored at the address currently held in CAC, say $K + i$. Next CAC is changed to $L + i$, and the contents of the k-register are stored. Then CAC is set back to $K + i$. After execution of these stores, $FF1$ is set to 1.

2. When F wants to fetch a pair of new values from the memory, the CAC is decreased by 1 if $FF1 = 0$, and is left unchanged if $FF1 = 1$. The address (con-

tents of CAC) is forced into the F Memory Address Register as a consequence of recognition of a special instruction in F by the Supervisory Control. Both the value n and the corresponding contents of the k-register are then fetched by the previously described $K - L$ interchange, and CAC is set back to $K + i$. At the end of the fetching operations, $FF1$ is set to 0.

Thus F always handles first the latest information brought in from V. If at any time CAC holds the address $K + M$, and if $FF1 = 1$, then $FF2$ is set, which prevents V from storing into the memory. (Of course for sufficiently large l, such an occurrence is very rare.) $FF2$ is reset by the resetting signal of $FF1$. If at any time CAC holds the address K, and if $FF1 = 0$, then $FF3$ is set, which prevents F from fetching. $FF3$ is reset by the setting signal of $FF1$.

Preliminary studies of this mode of transfer indicate that transfer of the first word takes at most two cycles, and the second takes one cycle. Thus the transfer of n and the contents of the k-register from V to F requires approximately 7 μ sec.

Department of Mathematics
Princeton University
Princeton, New Jersey

Department of Engineering
University of California
Los Angeles 24, California

Department of Mathematics
University of Oregon
Eugene, Oregon

Department of Engineering
University of California
Los Angeles 24, California

→ D. H. LEHMER, "A photo-electric number sieve," *Amer. Math. Monthly*, v. 40, 1933, p. 401–406.

2. D. H. LEHMER, "A machine for combining sets of linear congruences," *Math. Ann.* v. 109, 1934, p. 661–667.

3. D. H. LEHMER, "The sieve problem for all-purpose computers," *MTAC*, v. 7, 1953, p. 6–14.

4. D. H. LEHMER, "Teaching combinatorial tricks to a computer," *Proceedings of Symposia in Applied Mathematics*, v. X, American Mathematical Society, Providence, R.I., 1960, p. 179–193.

5. D. H. LEHMER, private communication, June 1960.

6. G. ESTRIN, "Organization of computer systems—the fixed plus variable structure computer," *Proceedings of the Western Joint Computer Conference*, May 1960, p. 33–37.

7. A. SVOBODA, "Rational numerical system of residual classes," *Stroje Na Zpracovani Informaci*. (Czechoslovakia), v. 5, 1957, p. 9–47.