NEW IDEAS FOR FACTORING LARGE INTEGERS

Carl Pomerance,[1] J. W. Smith[1] and S. S. Wagstaff, Jr.[1]

This is an extended abstract which summarizes papers [3], [4], and [5]. They describe improvements in the continued fraction factorization algorithm (CFRAC) and a special processor designed to execute this algorithm swiftly. The ideas in these papers will permit one to factor a 100 decimal digit integer in about a month on a processor which would cost about $1,000,000. Therefore, moduli for RSA cryptosystems should be chosen somewhat larger than 100 digits to be secure.

1. *Improvement of the CFRAC algorithm*

Let N be a large composite number. The CFRAC algorithm generates several sequences of integers derived from the simple continued fraction expansion of $\sqrt{N}$. See Morrison and Brillhart [1] for a detailed description of the algorithm. We need consider only the two sequences $\{Q_n\}$ and $\{A_n \bmod N\}$. They satisfy $|Q_n| < 2\sqrt{N}$ and $A_n^2 \equiv Q_n \bmod N$. The Q_n are unusually small quadratic residues modulo N. Since they are so small, many Q_n can be factored easily into primes. In the first step of CFRAC, the sequences are generated and some effort is made to factor the Q_n's by trial division by small primes. The Q_n's which can be factored, together with the corresponding A_n's, are saved on secondary storage. In the second step of CFRAC, the pairs (A_n, Q_n) saved in the first step are combined to construct a factorization of N. The first step requires many computer cycles but little memory, while the second step uses much memory for a short time. The two steps are run as separate jobs unless

1. Research supported in part by grants from the National Science Foundation.

N is small. The second step is now standard and easy. We offer no improvement on it. See Wunderlich and Parkinson [6] for recent work.

Consider the first step of CFRAC. The attempt to factor each Q_n involves trial division of Q_n by a set of small primes $p_1,...,p_m$ called the "factor base". The inner loop of the algorithm is:

```
Q' ← Q_n
i ← 1
while (i ≤ m) do
    if (Q' MOD p_i = 0) then
        Q' ← Q'/p_i
    else
        i ← i+1
if (Q' = 1) then output A_n, Q_n.
```

This loop is the computational bottleneck of the algorithm.

Most Q_n do not factor completely over the factor base. When $1 < Q' < p_m^2$ at the end of the inner loop, Q' must be a prime larger than p_m. In this case Q_n has been factored completely. Morrison and Brillhart noted [1] that such a Q_n can be used in the second step provided it can be paired with another Q_k having the same large prime Q'. Store Q' in the same record with A_n and Q_n. The second step begins by sorting the records in order of Q'. It then discards those records whose Q' is not repeated. This technique of using Q' is called the "large prime variation". It makes little difference in the asymptotic running time, but it halves the time required for numbers of practical size.

Another variation is the "early abort strategy" (EAS). (See Pomerance [2].) Choose $1 < m_1 < m$ and $2\sqrt{N} > B_1 > 1$. Run the inner loop up to $i = m_1$. At this point Q' is the largest divisor of Q_n all of whose prime factors exceed p_{m_1}. If $Q' > B_1$, abandon trial division of Q_n and obtain Q_{n+1}. One can use several early aborts by defining parameters

$$1 < m_1 < m_2 < \ldots < m \text{ and } 2\sqrt{N} > B_1 > B_2 > \ldots > 1.$$

Each time i reaches the next m_j a decision is made: the inner loop is broken if $Q' > B_j$. Optimal asymptotic choices for the m_j and B_j are given in [2] where it is shown that EAS produces a large reduction in asymptotic running time. The time needed to factor a typical 50 digit number on an IBM 370/158 is about 100 hours when EAS is not used, 30 hours with one abort, 14 hours with two aborts, and 12 hours with three

aborts. (These times assume that the large prime variation is used, too.) Good practical choices for the EAS parameters were found by experimentation and are given in [3]. With one abort choose $m_1 = 50$ and $B_1 = \sqrt{N}/1000000$ for $m = 959$ and $10^{40} < N < 10^{54}$. The asymptotic running time of CFRAC with EAS is about $L(N)^{1.23}$, where $L(N) = \exp((\ln N \ln \ln N)^{1/2})$. (The time is about $L(N)^{1.41}$ for CFRAC without EAS.)

The numbers A_n are computed by the congruence $A_n \equiv q_n A_{n-1} + A_{n-2} \bmod N$. Since reduction modulo N is expensive and the positive integers q_n are usually small ($q_n = 1$ 42% of the time), the remaindering is performed only occasionally. We recommend doing it only when A_n approaches N^2, q_n exceeds 10^6, or Q_n has been factored so that it and $A_n \bmod N$ must be output.

2. *Special hardware for factoring large numbers*

The CFRAC algorithm is amenable to parallel computation. One can divide a single Q_n by many primes at once, or many Q_n's by one prime at once, or use a hybrid of the methods. The best choice depends on the machine's architecture. One must choose new EAS parameters for each architecture. One can generate the Q_n's (and other sequences) in parallel because there is a way to jump far ahead in the continued fraction expansion. Wunderlich is programming CFRAC on several parallel machines such as the DAP and the MPP.

We decided to design and build a special processor for factoring large integers by CFRAC. With a grant from the University of Georgia Faculty Research Grants Program in 1982, we hoped to design a machine which could factor a 70 digit number in a few days. Our small budget allowed only a tiny memory. Therefore, we planned a machine to perform just the first step of CFRAC. It would communicate with a host minicomputer capable of doing the second step. The fabrication of this special processor has just been completed at this writing (November, 1983). We expect it to factor large numbers soon.

Although we had limited resources, we gave our machine several computational accelerators. Most of the CFRAC arithmetic involves numbers as large as $2\sqrt{N}$. The 128-bit operand size of the processor permits it to factor numbers as large as 76 digits via CFRAC. This feature gives the machine the name Extended Precision Operand Computer or EPOC. It is often playfully called the "Georgia Cracker". The word size could be extended even further without great difficulty. We chose 128 bits because of the speed of the EPOC. It will take several months to factor a 76 digit number. A larger word size would have wasted capacity. A smaller word size would have prevented us from factoring numbers as large as feasible in a reasonable time.

Another accelerator provides rudimentary parallel processing. The remaindering operations of the inner loop are executed in parallel by a separate unit of the EPOC. A set of modulus elements (dubbed the "mod squad") divides one Q_n by several different primes at once. The elements are loaded with the primes and then the dividend Q_n is broadcast to all of them one bit at a time. The mod squad reports a bit vector which identifies those primes which divide Q_n exactly. During the parallel remaindering the main processor finds Q_{n+1} or makes the EAS decisions.

As the main processor performs the EAS tests, it modifies the EAS parameters occasionally to keep itself and the remaindering units busy as much as possible. When Q_n is factored, A_n, Q_n, and the large prime (if any) are transmitted to the host computer for storage. This action keeps the EPOC memory requirements low. Communication between EPOC and the host is performed by the input/output terminal emulator (IOTE). Because CFRAC for $N > 10^{50}$ is compute-bound the IOTE can be relatively slow. It has a DMA interface to the EPOC, but appears to be a 9600 baud terminal to the host.

The EPOC language is horizontal microcode. Each data bus is controlled by the programmer during each instruction cycle. The source program is prepared on the host computer. It is assembled, linked, and loaded by systems programs executing on the host. The assembler is a general two-pass cross-assembler driven by a language definition program. Results from the EPOC are moved to the host by the IOTE and a host unloader program. Several diagnostic programs are run occasionally to insure correct functioning of the EPOC. The IOTE is connected to a console which allows the operator to monitor the EPOC calculations. Relations such as $A_n^2 \equiv Q_n \pmod N$ are used by the CFRAC program to check for hardware failures. Most of the systems programs are written in RATFOR for portability.

The EPOC is constructed in Schottky TTL technology using a bit-slice architecture. This technology combines reasonable speed with simplicity of design. The multibus prototype cards were wirewrapped by a machine using a wire list produced by computer-aided design. The 128-bit ALU is mounted on four cards (32 bits per card). The IOTE and a sequencer each occupy one card. The remainder units are packaged separately. There are about 10 of them.

The machine described so far will likely factor a 70 digit number in about two months. We believe that the following enhancements will permit us to build another processor which will be able to factor a 100 digit number in a few years. We are designing a VLSI chip to do the job of a remaindering unit, which occupies one card in the present EPOC. With this chip one can have hundreds of remainder units in one EPOC. This amount of dividing power will force reconsideration of the EAS

parameter choices. The factor base will have to be enlarged. Generation of the continued fraction expansion will become a significant part of the whole calculation. We may build several main processors to share one VLSI mod squad. The main processor(s) will be fabricated in ecl technology and will have wider operands. Because of the parallelism possible in the algorithm a few dozen of these processors could factor a 100 digit number in about a month at a cost of about $1,000,000 for the machine.

REFERENCES

1. M. A. Morrison and J. Brillhart, A method of factoring and the factorization of F_7, *Math. Comp.* 29 (1975), 183-205.
2. C. Pomerance, Analysis and comparison of some integer factoring algorithms, in *Computational Methods in Number Theory,* Part 1, H. W. Lenstra, Jr. and R. Tijdeman, eds., Math. Centrum 154, Amsterdam (1982), 89-139.
3. C. Pomerance and S. S. Wagstaff, Jr., Implementation of the continued fraction integer factoring algorithm, *Congressus Numerantium* 37 (1983), 99-118.
4. J. W. Smith and S. S. Wagstaff, Jr., An extended precision operand computer, *Proceedings of the 21st Southeast Region ACM Conference* (1983), 209-216.
5. J. W. Smith and S. S. Wagstaff, Jr., How to crack an RSA cryptosystem, to appear in *Congressus Numerantium.*
6. M. C. Wunderlich and D. Parkinson, A memory-efficient algorithm for Gaussian elimination over GF(2) implemented on highly parallel computers, in preparation.