05-1D

The 2nd International Conference on Cybernetics and Information Technologies, Systems and Applications

11th International Conference on
Information Systems Analysis and Synthesis

July 14-17, 2005 Orlando, Florida ~ USA

PROCEEDINGS

Volume II

Edited by
Jose Aguilar
Hsing-Wei Chu
Elena D. Gugiu
Ilka Miloucheva
Naphtali Rishe

Organized by
International Institute of Informatics and Systemics
IIIS
Member of the International Federation of Systems Research (IFSR)

Information Defense Technology on the Base of Modular Algebra

Oleg D. JOUKOV
Moscow State University, Vorobyevy Gory
Moscow, 119899, Russia

Naphtali D. RISHE, Andriy B. SELIVONENKO
School of Computer Science, Florida International University
Miami, Florida 33199, USA

Carlos CANAS, Kiesha PIERRE, Asha BRITO
Division of Computer Sciences and Mathematics, Florida Memorial University
15800 NW 42nd Ave Miami Gardens, FL 33054, USA

ABSTRACT

An intensification of information processes in modern society creates some serious problems - information security, for example. In many cases it is impossible to say about usefulness of information without efficient deciding this problem. This paper discusses a possibility of creating a high performance computing technology for information defense, processing very long integers and based on a combination of modular algebra and methods, used, for example, in cryptography.

Keywords: modular algebra, cryptography, electronic signature

1. INTRODUCTION

Modular computations plays important role in information security systems (IDS), for example, in cryptography and electronic signature technology [1]. The paper is devoted to this direction. The research is based on the method presented in [2] and number theory, in particular, remainder systems (RS). These systems provide a good means for extremely long integer algebra. Their carry-free operations make parallel implementation feasible. Some applications involving very long integers, such as public key encryption, rely heavily on fast modulo reductions.

The power of the method [2], is shown on an example of the operation $x = x^y \bmod z$ or (for other denoting) $x = |x^y|_z$, where x,y,z are integer in the order of several hundreds digits. In the paper it is discussed a possibility of speeding up algorithm, presented in [1] and used, for example, in cryptography and other applications. Before describing the presented technology let us recall main features of RS.

Given a set of relatively prime moduli,

$(m_1,m_2,...,m_n)$; $\gcd(m_i,m_j)=1$; $i\neq j$.

The base M of RS equals to the product $M=\prod_{i=1}^{n} m_i$. All numbers A in RS have a unique representation given by the tuple $A=(a_1,a_2,...,a_n)$ where $a_i=|A|_{m_i}$; $i=1,...,n$. Here a_i is called a remainder of A modulo m_i.

According to the Chinese Remainder Theorem (CRT) [3] A can be computed on the base of the following expression:

$$A=\left|\sum_{i=1}^{n} a_i Q_i |Q_i^{-1}|_{m_i}\right|_M \qquad (1)$$

where $Q_i=M/m_i=(0,0,...,q_i,...,0)$ and $|Q_i^{-1}|_{m_i}$ denotes the multiplicative inverse of $|Q_i|_{m_i}$.

Contrary to the more common, fixed-radix, weighted number representation, RS does not suffer from carry propagation. Addition, subtraction, and multiplication can be done in parallel and constant time for all remainders of operands. All digits are computable independently, and thus concurrently.

Comparison, sign detection, overflow detection, and scaling can be achieved through an procedure known as base extension (BE). With base extension the base of the system, $(m_1,m_2,...,m_n)$, can be extended to a larger set

$(m_1,m_2,...,m_n,m_{n+1},m_{n+2},...,m_l)$

with all m_i relatively prime. It is then necessary to compute

$a_1,a_2,...,a_n,a_{n+1},a_{n+2},...,a_l$

from the given n tuple $(a_1,a_2,...,a_n)$. This procedure can be done in various ways but we shall use the following method more suitable for our case. The method approaches the solution through an approximation which is correct in nearly all cases. The rare occurrences of uncertainties about the correctness of the result are detected, and corrected with additional approximations. Starting with a modification of (1),

$$A=\sum_{i=1}^{n} a_i Q_i |Q_i^{-1}|_{m_i} - hM \qquad (2)$$

all a_j $(j=n+1,...,l)$ can be computed as

$$\sum_{i=1}^{n} a_i Q_i |Q_i^{-1}|_{m_i} |m_j|_{m_j} - |hM|_{m_j} |m_j|_{m_j}. \qquad (3)$$

The only variable unknown in (3) is h. For this method

$$H=h+h'=\sum_{i=1}^{n} a_i|Q_i^{-1}|_{m_i}/m_i,$$

where $h' < 1$.

In real implementations, limited precision of the representation of the rational factors $Q_i^{-1}/m_i < 1$ causes problems when the exact value of H comes close to some integer. This situation is easily detected by observing a definable amount of ones in the binary representation of the fractional part h' of H. As it's seems additional approximations can clearly identify the correct value of h. Luckily, this situation is of no importance in the context of the main topic of this paper. No additional approximations are needed at all. As it will be shown, it does not matter, whether h is computed correctly, or off by 1.
The underlying implementation of RS arithmetic in this paper assumes l processing elements (PE), or sets of PEs, working in parallel. Each PE is dedicated to some mi and independently computes

$a_i=|x_iy_i+z_i|_{mi}$.

All PEs are grouped in two sets, RS1=($m_1,m_2,...m_n$), and RS2 = ($m_{n+1},...,m_l$). Each base extension consumes the time of broadcasting n numbers from one set to the other, and l n numbers-vice versa. For the sake of simplicity a bus coupled system is assumed. Then, base extension consumes n or l n bus cycles respectively.

The method proposed in [2] is based on the two congruences:

$$A-|A|_M|S^{-1}|_MS \equiv |A|_S, \quad (4)$$

and

$$A-|A|_M|S^{-1}|_MS \equiv |0|_M. \quad (5)$$

The congruence (4) can be computed as it is shown in [3]. The key issue is to choose M such that $|A|_M$ trivially. For fixed-radix, weighted number systems M is typically some integer power of the radix. But for RS representation $M=\prod_{i=1}^{n} m_i$, is preferable. Then, $|A|_M \equiv A$.

The congruences given in (4) and (5) can be derived by the following observations

$$A-|A|_M=tM \equiv |0|_M$$

with A,t,M integer. Take some S with gcd(S,M)=1. Then the multiplicative inverse $|S^{-1}|_M$ exists. Combination of the two arguments given above leads to:

$$|A-|A|_M|S^{-1}|_MS|_M \equiv 0,$$

where $|S^{-1}|_MS|_M \equiv 1$, and

$$|A-|A|_M|S^{-1}|_MS|_S \equiv |A|_S,$$

where $|A|_M|S^{-1}|_M$ is integer.

Evidently, $|S^{-1}|_M$ is a pre-computed constant. Thus, a value of $|A|_S$ can be computed with two multiplications, one addition, and division of the result by M. The final result is a smaller number $|AM^{-1}|_S$. k iterations of this basic process lead to some smaller number $|AM^{-k}|_S$.

2. REALIZING METHOD BASED ON RS

In the case of using RS it is preferable to take M equal to a base of RS ($M=\prod_{i=1}^{n} m_i$). Then $|A|_M \equiv A$. But there is requested for a larger RS base for representing numbers larger than M–1. Additional digits (remainders) in RS with a larger base have to be determined by procedure of extending the source base by introducing additional moduli.
Correct computing $z=|AB|_S$ in RS1 with moduli ($m_1,m_2,...,m_n$)

$$y=x/M_1=(x-|x|_{M1})M^{-1}$$

and in RS2 moduli ($m_{n+1},m_{n+2},...,m_l$) is possible under a condition $A,B,S < M_1 < M_2$, where $M_1=\prod_{i=1}^{n} m_i$ and $M_2=\prod_{i=n+1}^{l} m_i$.

Then the product $AB < M_1M_2$ can be represented in an extended RS with the base M_1M_2, and the result $x \equiv |AB|_S < M_1 < M_2$ can be used for next approximations. The integer division in RS is represented by next steps:

1) $x=AB < M_1M_2$ is the source dividend;
2) $|x|_{M1}$ corresponds to x in RS1;
3) In order to compute $x-|x|_{M1}$, which can be larger than M_1, a base extension from RS1 to RS2 is necessary. In RS1 $x-|x|_{M1}$ has all zeroes.
4) $(x-|x|_{M1})M_1^{-1}$ is small enough to be represented in RS2 only. The division by M_1 has no remainder and can be substituted by multiplying with the inverse M_1^{-1}. This inverse does not exist in RS1. This is due to the fact that $\gcd(M_1,M_1) \neq 1$ which is a necessary condition for the existence of the inverse.
5) With a final base extension back to RS1 results in both bases are available again.

The method for base extension is done in two phases. In a first phase an approximation for h is computed. This approximation gives the correct result in most cases. But in rare occurrences the approximation is off by 1.

In this section it is shown a slightly modified algorithm for modulus reduction in RS1. It works in any case, no matter whether the approximated value of h is correct or off by 1. Thus, base extension is substantially simplified. The algorithm represented below produce $f=|AB/M_1|_S$, where A and B are source numbers. Next steps of the algorithm of the method are performed in RS1 and RS2 respectively.
Step 1
RS1: x=AB; multiplying
Step 2
RS1: $x \equiv |x|_{M1}$; representation x in RS1
Step 3
RS1: $g=xS^{-1}$; multiplying
Step 4
RS1 and RS2: g- > [g~]; approximate extension from RS1 to RS2
Step 5
RS2: k=[g~]S; multiplying
Step 6
RS2: d=x–k; subtraction

Step 7
RS2: $y=dM_1^{-1}+2S$; multiplying and adding
Step 8
RS1 and RS2: $[y\sim] < -y$; return approximate extension RS2 to RS1
Step 9
$t=4h'$; determining appropriate value t
Step 10
RS1: $f=[y\sim] - tS$; multiplying and subtraction
RS2: $f=y-tS$; multiplying and subtraction
For the correctness of the algorithm, the following assumptions have to be made
$S+\delta_S < M_1 < S+S/3$ with $\delta_S < S/6$;
$4S \le M_2 \le (4+e_S)S$ with $e_S < 1/12$, where e_S– error in M_2/S;
$A,B < S+\delta_S < S+S/6$.
These assumptions do not put any burden on the applicability of the algorithm if S is large enough and anyway it is to be larger or equal to $M_2/(4+\delta_S)$.

3. ANALYZING STEPS AND CONDITIONS OF THE ALGORITHM

1) $x=AB$: The product AB can be represented in the overall RS (ORS), based on M1 and M2, and meeting to the condition

$$x=AB < S^2+2S/6+S^2/36 < M_1M_2.$$

2) $g=||x|_{M1}S^{-1}|_{M1}$: According to the method [6] the first step is to compute $|x|_{M1}$. In RS1 it is trivial, as $|x|_{M1} \equiv x$. The equivalent digits in RS2 are not needed at all. The same argument holds for

$$g=||x|_{M1}S^{-1}|_{M1} < M_1.$$

3) Base extension from RS1 Q B1 to RS2: The next step, multiplication with S, possibly results in a product larger than M1. Thus, first a base extension from RS1 to RS2 has to take place. As indicated ealier an approximation for the real value of h is sufficient

$$\tilde{h}=[\sum_{i=1}^{n} x_i \tilde{v}_i],$$

where $[v\sim]=v-e_i$ and $e_i << 1$ denoting the error between the real value $v_i=Q_i^{-1}/m_i$ and its representation in a limited precision format.
Base extension from RS1 to RS2 results in some $[g\sim]$ on the basis of (2)
$[g\sim]=g$, if $[h\sim]=h$ (correct approximation),
$[g\sim]=g+M_1$, if $[h\sim]=h-1$ (approximation off by 1).
The range of possible values $[g\sim]$ varies as
$[g\sim] < M_1$, if $[h\sim]=h$,
$[g\sim]=M_1+\delta_S$, if $[h\sim]=h-1$.
4) $k=(||x|_{M1}S^{-1}|_{M1})S$: According the method [2], $k=[g\sim]S$ has to be computed next.
5) $y=(x-||x|_{M1}S^{-1}|_{M1}S)M_1^{-1}+2S$: as $|x-k|_{M1} \equiv 0$, it can be divided by M_1 without a remainder. An ajuistment with 2S is necessary in order to avoid negative values for y:

$$y=(x-k)M_1^{-1}+2S.$$

Then $S/2 < y < 3S+S/3$. Above result can be obtained by checking the ranges of all subresults of y:

$$0 \le x < (S+\delta_S)^2 < M_1^2;$$

$$-(S+S/3+S/6)S < x-\tilde{g}\,S < M_1^2;$$

$$-3S/2 < (x-\tilde{g}\,S)M_1^{-1} < M_1 < S+S/3;$$

$$S/2 < y < 3S+S/3;$$

where $S+S/3 > M1$ and $S/6 > \delta_S$.
6) Base extension from RS2 to RS1: Despite the fact that y might not fit into the range of RS1, a base extension is done here. Necessary adjustments- subtractions of suitable multiples of S-follow in the next step. Notice
if $[y\sim]=y$ a result is correct;
if $[y\sim]=y+tS$ adjustments are necessary.
For base extension a value $[h\sim]$ is computed and will be in the following steps.
7) For RS1: $[f\sim]=[y\sim]$ tS; for RS2: $f=y$ tS.
In this step a check must be made whether y is representable in RS1. It is very likely that $y \ge S+\delta_S$. But only a value less than $S+\delta_S$, could really be used for further modulo multiplications. Therefore, a suitable integer multiple, t, of S must be subtracted from y and $[y\sim]$
$f=y-tS < S+\delta_S$ in RS2 and
$f=[y\sim]-tS < S+\delta_S$ in RS1.
8) $t=4h'$: From (2) follows that the fractional part of H equals to $h'=y/M_2 < 1$.
As to the relation between y and h' notice the following. Scaling of h' with the factor M_2/S would result in integer values at integer multiples of S. The value $[h'M_2/S]$ indicates how many times S has to be subtracted from y in order to get a value $f=y$ tS $< S < M_1$. Thus, f can be represented in RS1.
But all computational errors due to limited precision have to be taken into account. This becomes crucial as t results from a truncation of $h\,M_2/S$.
1) The computed value $[h\sim]'$ is very likely smaller than the real value h', as all values $\tilde{v}_i$ ($v_i=|Q_iQ_i^{-1}|_{mi}$) are represented with limited precision

$$\tilde{h}' > h'-e_H,$$

with e_H denoting the maximum possible error.
2) The computed value $[r\sim]_S=(M_2/S)$ has also some error e_S relatively to r_S: $[r\sim]_S-e_S$.
3) The error of t (e_t) is

$$e_t=(h'M_2/S \quad 0 \le \tilde{g} < (M_1+\delta_S) \quad {}_S-e_He_S < 5e_H+e_S.$$

The error e_t becomes important for values $h'r_S=h'M_2/S$ slightly larger than some integer. Then, truncation could result in a wrong value of t. This can only happen for values of y slightly larger than some integer multiple of S. If $e_t < \delta_S/S < 1/6$, then uncertainties about the value of t can only happen for y in the intervals

$$[S,S+\delta_S];[2S,2S+\delta_S];[3S,3S+\delta_S].$$

A "wrong" t would then result in a f in the interval $[S,S+\delta_S]$. But values in this interval are allowed for subsequent modulo multiplications, since initially the assumption $0 \le A < S+\delta_S$ was made.

An equal distribution of the total error, e_t, among the components leads to $e_H < 1/60$ and $e_S < 1/12$.

Then, $e_t < 5e_H+e_S < 1/6$ and r_S is chosen to be slightly larger than 4: $4+e_S$. In this case can be used the value $[r\sim]_S=4=2^2$. Notice, this comes in handy, as multiplication by some integer power of 2 is simple.

4. CONCLUSION

This paper shows a possibility of the combination remainder systems with the efficient modulo reduction method [2]. Typical sizes of numbers, for example, for information technologies of obtaining digital signatures and public-key cryptosystems are in the order of 2^{600}–2^{700} and larger. If processing elements (PE) with 32 bit word size are used, approximately 20 PEs make up a system which is able to compute numbers of this order. Although commercially available PEs can be employed, specialized PEs can be tuned to much higher speeds.

Each PE is dedicated to some mi with all mi relatively prime. In order to exploit processing capabilities of the PEs, mi is chosen to be close to, but less than 232. It can be shown that under the restrictions given above-20 PEs with different mi s, mi < 232 the typical cryptography operation AB+C mi approximately needs the time of (32+11+12) additions. Taking this time as one computational cycle, the method [2] realized in RS needs 5 such cycles plus 3 base extensions.

Base extension in essence is a summation process of the same kind as shown above. First, the sum

$$\sum_{i=1}^{n} x_i|Q_iQ_i^{-1}|_{mj}|_{mj},$$

is computed in every j–th PE by broadcasting all x_i. At the same time, an additional PE computes $[H\sim]=\sum_{i=1}^{n} x_i[v\sim]_i$ in a similar way which is broadcast to all other PEs afterwards. Then it is multiplied with $|-M|_{mj}$. A final addition in each PE produces x_j.

Assuming a bus-coupled net of PEs, base extension can be performed in approximately n cycles. Each cycle involves broadcasting of some x_j, and/or computing $|AB+C|_{mj}$.

In conclusion it is necessary to notice, that the combination of the method similar to [2] with RS gives a possibility of processing computer words of the length 32 bits instead of 600-700 (and more) bits.

5. ACKNOWLEDGEMENTS

This material is based on work supported by NATO under Grant No. SST.NR.CLG:G980822 and by the National Science Foundation under Grants No HRD-0317692, EIA-0320956, and EIA-0220562.

6. REFERENCES

[1] Rivest R., Shamir A., and Adleman L., **A method for obtaining digital signatures and public-key cryptosystems**, Commun. ACM, 1978, pp. 120-126, 1978.

[2] Montgomery P.L., **Modular multiplication without trial division**, Mathemat. Comput., Vol. 44, No. 170, 1985, pp. 519-521.