RNS bases and conversions

Jean-Claude Bajard
Thomas Plantard
-
SPIE 2004

LIRMM - Montpellier - France

Contents

Context

Why RNS?

- Distribute operations from huge number to small residues

RNS conversions

- Modular multiplication
- Division

Context

Why RNS?

- Distribute operations from huge number to small residues

RNS conversions

- Modular multiplication
- Division

The different conversions

- Radix $\beta \leftrightarrows$ RNS (punctual uses)
- RNS $\longrightarrow$ RNS (modular multiplication, division..)

Chinese Remainder Theorem

Chinese Remainder Theorem

- A RNS basis is $(m_1, m_2, \ldots, m_n)$ with $M = \prod_{i=1}^{n} m_i$
- If we consider $(x_1, x_2, \ldots, x_n)$ with $0 \leq x_i < m_i$
- We have $\exists! X < M$ with $x_i = |X|_{m_i} = X \bmod m_i$

Chinese Remainder Theorem

Chinese Remainder Theorem

- A RNS basis is $(m_1, m_2, \ldots, m_n)$ with $M = \prod_{i=1}^{n} m_i$
- If we consider $(x_1, x_2, \ldots, x_n)$ with $0 \le x_i < m_i$
- We have $\exists! X < M$ with $x_i = |X|_{m_i} = X \bmod m_i$

Properties

- Advantage: Addition and multiplication can be parallelized.
- Drawback: Comparison and division are difficult.

Chinese Remainder Theorem

Chinese Remainder Theorem

- A RNS basis is $(m_1, m_2, \dots, m_n)$ with $M = \prod_{i=1}^{n} m_i$
- If we consider $(x_1, x_2, \dots, x_n)$ with $0 \le x_i < m_i$
- We have $\exists! X < M$ with $x_i = |X|_{m_i} = X \bmod m_i$

Properties

- Advantage: Addition and multiplication can be parallelized.
- Drawback: Comparison and division are difficult.

Example

- A RNS base: $(255, 256, 257)$ with $M = 16776960$
- $a = 10000 \longrightarrow (55, 16, 234)$ and $b = 300 \longrightarrow (45, 44, 43)$
- $a \times b = 3000000 \longrightarrow (180, 192, 39)$

Conversion: From CRT

From the Chinese Remainder Theorem

$$X = \left| x_1 \left|M_1\right|_{m_1}^{-1} M_1 + x_2 \left|M_2\right|_{m_2}^{-1} M_2 + \ldots + x_n \left|M_n\right|_{m_n}^{-1} M_n \right|_M \quad (1)$$

Where, $M_i = \dfrac{M}{m_i}$ and $\left|M_i\right|_{m_i}^{-1}$ represents the inverse of M_i modulo m_i .

Conversion: From CRT

From the Chinese Remainder Theorem

$$X = \left| x_1 \left| M_1 \right|_{m_1}^{-1} M_1 + x_2 \left| M_2 \right|_{m_2}^{-1} M_2 + \ldots + x_n \left| M_n \right|_{m_n}^{-1} M_n \right|_M \quad (1)$$

Where, $M_i = \dfrac{M}{m_i}$ and $\left| M_i \right|_{m_i}^{-1}$ represents the inverse of M_i modulo m_i .

Find α

$$X + \alpha M = \left(\sum_{i=1}^{n} \alpha_i M_i \right) \quad (2)$$

- Shenoy Kumaresan, in 1989, proposed to use an extra modulo
- Posch and Posch, in 1995, proposed a floating point approach

Conversion: Via MRS

A Mixed Radix System

- We can represent $X < M$ with $(x'_1, x'_2, \ldots, x'_n)$ with $0 \leq x'_i < m_i$

$$X = x'_1 + x'_2 m_1 + x'_3 m_1 m_2 + \ldots + x'_n m_1 \ldots m_{n-1} \quad (3)$$

Conversion: Via MRS

A Mixed Radix System

- We can represent $X < M$ with $(x_1', x_2', \ldots, x_n')$ with $0 \leq x_i' < m_i$

$$X = {x'}_1 + {x'}_2 m_1 + {x'}_3 m_1 m_2 + \ldots + {x'}_n m_1 \ldots m_{n-1} \qquad (3)$$

Parallel algorithm

$$\begin{cases} x_1' = x_1 \bmod m_1 \\ x_2' = (x_2 - x_1')m_{1,2}^{-1} \bmod m_2 \\ x_3' = ((x_3 - x_1')m_{1,3}^{-1} - x_2')m_{2,3}^{-1} \bmod m_3 \\ \vdots \\ x_n' = (\cdots(x_n - x_1')m_{1,n}^{-1} - x_2')m_{2,n}^{-1}) - \cdots - x_{n-1}')m_{n-1,n}^{-1} \bmod m_n \end{cases}$$

Conversion: Via MRS

A Mixed Radix System

- We can represent $X < M$ with $(x'_1, x'_2, \ldots, x'_n)$ with $0 \leq x'_i < m_i$

$$X = {x'}_1 + {x'}_2 m_1 + {x'}_3 m_1 m_2 + \ldots + {x'}_n m_1 \ldots m_{n-1} \quad (3)$$

Sequential algorithm

$$\begin{cases} x'_1 = x_1 \bmod m_1 \\ x'_2 = (x_2 - x'_1)\, |m_2|^{-1}_{m_2} \bmod m_2 \\ x'_3 = ((x_3 - x'_1) - x'_2 m_1)\, |m_1 m_2|^{-1}_{m_3} \bmod m_3 \\ \vdots \\ x'_n = ((x_n - x'_1) - m_1(x'_2 - m_2(x'_3 - \cdots - m_{n-3}(x'_{n-2} - m_{n-2}x'_{n-1})\ldots) \end{cases}$$

Criteria for conversion

CRT

- Find α
- Use M_i and $|M_i|_{m_j}^{-1}$ which are difficult to characterize.

Criteria for conversion

CRT

- Find α
- Use M_i and $|M_i|_{m_j}^{-1}$ which are difficult to characterize.

MRS

- A parallelize method: $n^2/2$ modular divisions by specific numbers.
- Sequential method: $n^2/2$ products by characterized numbers and n modular divisions by non particular numbers.

Remark about MRS to new RNS

From MRS to new RNS: using Horner

$$X \bmod \widetilde{m_j} = [x'_1 + (m_1 - \widetilde{m_j})(x'_2 + (m_2 - \widetilde{m_j})(x'_3 + \ldots + (m_{n-1} - \widetilde{m_j})x'_n)\ldots)] \tag{6}$$

Remark about MRS to new RNS

From MRS to new RNS: using Horner

$$X \bmod \widetilde{m_j} = [x'_1+(m_1-\widetilde{m_j})(x'_2+(m_2-\widetilde{m_j})(x'_3+\ldots+(m_{n-1}-\widetilde{m_j})x'_n)\ldots)] \tag{6}$$

Proposition for a basis: Minimization of those differences

$\widehat{d}$	2^8	2^{10}	2^8	2^{10}	2^8	2^{10}
m	2^{32}	2^{32}	2^{64}	2^{64}	2^{128}	2^{128}
$\frac{\widehat{d}}{\log(\widehat{d})}$	46	147	46	147	46	147
# coprimes found	39	117	44	124	41	121

Table: coprimes found between m and $m+d$ with a trivial algorithm

Criteria for conversion

MRS sequential

- $n^2/2$ products by small numbers
- n classic modular divisions

MRS parallelize

- $n^2/2$ modular divisions by small numbers

Modular Division

Idea

- We want to compute $y = x \times |d|_m^{-1} \bmod m$
- Montgomery return a residue of the product by an inverse

Modular Division

Idea

- We want to compute $y = x \times |d|_m^{-1} \bmod m$
- Montgomery return a residue of the product by an inverse

Modular Division

1. Input: An modulo m, a divisor d and a variable x
2. Precompute:
   1. $m = r_m + q_m d$ with $r_m < d$
   2. $I_m = (-m)^{-1} \bmod d$
3. Algorithm:
   1. $x = r_x + q_x \times d$ with $0 \leq r_x < d$
   2. $k \leftarrow r_x \times I_x \bmod d$
   3. $y \leftarrow (r_x + k \times r_m)/d + q_x + k \times q_m$ $\quad (\sim y = \dfrac{x - km}{d})$
4. Output: $y = x \times d^{-1} \bmod m$

Example

Classic Modular Division

1. Input: $m = 10007$, $d = 15$, $x = 7856$
2. Precompute: $(d^{-1} \bmod M) = 4670$
3. Algorithm: $s = x \times (d^{-1}) \bmod M$
   - $s = 7856 \times 4670 = 36687520$
   - $s = 36687520 \bmod 10007 = 1858$

Example

Classic Modular Division

1. Input: $m = 10007,\ d = 15,\ x = 7856$
2. Precompute: $(d^{-1} \bmod M) = 4670$
3. Algorithm: $s = x \times (d^{-1}) \bmod M$
   - $s = 7856 \times 4670 = 36687520$
   - $s = 36687520 \bmod 10007 = 1858$

Modular Division

1. Input: $m = 10007,\ d = 15,\ x = 7856$
2. Precompute: $r_m = 2,\ q_m = 667,\ I_m = 7$
3. Algorithm:
   1. $x = r_x + q_x \times d \longrightarrow r_x = 11, q_x = 523$
   2. $k \leftarrow 11 \times 7 \bmod 15 = 2$
   3. $y \leftarrow (11 + 2 \times 2)/15 + 523 + 2 \times 667 = 1858$

Example

Classic Modular Division

1. Input: $m = 10007$, $d = 15$, $x = 7856$
2. Precompute: $(d^{-1} \bmod M) = 4670$
3. Algorithm: $s = x \times (d^{-1}) \bmod M$
   - $s = 7856 \times 4670 = 36687520$
   - $s = 36687520 \bmod 10007 = 1858$

Modular Division

1. Input: $m = 10007$, $d = 15$, $x = 7856$
2. Precompute: $r_m = 2$, $q_m = 667$, $I_m = 7$
3. Algorithm:
   1. $x = r_x + q_x \times d \longrightarrow r_x = 11, q_x = 523$
   2. $k \leftarrow 11 \times 7 \bmod 15 = 2$
   3. $y \leftarrow (11 + 2 \times 2)/15 + 523 + 2 \times 667 = 1858$

Internal Reduction

Modular Reduction

1. Input: An integer $2^{\delta-1} \leq d < 2^{\delta}$ A variable $x < 2^t$
2. Algorithm:
   - $R \leftarrow x,\ Q \leftarrow 0, T \leftarrow t$
   - WHILE $T > \delta + 2$ DO
     1. Splitting: $R \longrightarrow R_1, R_0$
     2. Call: $(r_{R_1}, q_{R_1}) \leftarrow Barrett_k(R_1, d)$
     3. Updating: R, Q, T
   - WHILE $R \geq d$ DO $R \leftarrow R - d, Q \leftarrow Q + 1$
3. Output: $r_x \leftarrow R,\ q_x \leftarrow Q$

$Barrett_k$

Generalized Barrett Property

$$x - d\left\lfloor \frac{\left(\left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor \left\lfloor \frac{x}{2^{\delta-1}} \right\rfloor\right)}{2^{k+1}} \right\rfloor < 3d \tag{7}$$

$Barrett_k$

Generalized Barrett Property

$$x - d\left\lfloor \frac{\left(\left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor \left\lfloor \frac{x}{2^{\delta-1}} \right\rfloor\right)}{2^{k+1}} \right\rfloor < 3d \qquad (7)$$

$Barrett_k$

1. Input: $x < 2^{\delta+k}$ and $2^{\delta-1} \leq d < 2^{\delta}$
2. Precompute: $\left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor$
3. Algorithm:
   - $q \leftarrow \left(\left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor \left\lfloor \frac{x}{2^{\delta-1}} \right\rfloor\right)$
   - $q \leftarrow \left\lfloor \frac{q}{2^{k+1}} \right\rfloor$
   - $r \leftarrow x - qd \bmod 2^{\delta+1}$ (as , $r < 3d$)
4. Output: (q, r) such that $r = x - dq < 3d$

Analysis

Complexity of a modular division

1. Modular multiplication by the inverse: $2t^2 + 4t$
2. Our approach:
$$\left\lceil \frac{(t-\delta-2)}{(\delta-2)} \right\rceil (2\delta^2 + 3\delta + 4 + \log(t)) + 3\delta^2 + \delta t + 12\delta + 2t$$

Example of comparison

d	2^8	2^{10}	2^8	2^{10}	2^8	2^{10}
m	2^{32}	2^{32}	2^{64}	2^{64}	2^{128}	2^{128}
Our approach	1252	1521	2386	2868	4219	4234
Barrett algo	2176	2176	8448	8448	33280	33280

Table: Number of binary operations for a modular division

Conclusion

Conclusion

1. Moduli in small intervals are interesting.
2. This kind of RNS basis are easy to build.
3. MRS is a good choice with this type of moduli and with dedicate algorithms.
4. In future, we want to build complete operators for RNS crypto implementation.