RNS bases and conversions

Jean-Claude Bajard
Thomas Plantard

SPIE 2004

LIRMM - Montpellier - France
Contents

1 Introduction
   • Context
   • Residue Number System

2 Conversion
   • From the Chinese Remainder Theorem
   • Via Mixed Radix System
   • Criteria

3 Basis Selection Criteria
   • MRS to new RNS
   • Modular Division
   • Internal Reduction

4 Conclusion
   • Complexity
   • Perspective
Why RNS?

- Distribute operations from huge number to small residues

RNS conversions

- Modular multiplication
- Division
Why RNS?

- Distribute operations from huge number to small residues

RNS conversions

- Modular multiplication
- Division

The different conversions

- Radix $\beta \leftrightarrow$ RNS (punctual uses)
- RNS $\rightarrow$ RNS (modular multiplication, division..)
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 \(x_i = |X|_{m_i} = X \mod m_i\).
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 \(x_i = X \mod m_i\).

Properties

- Advantage: Addition and multiplication can be parallelized.
- Drawback: Comparison and division are difficult.
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 \(x_i = |X|_{m_i} = X \mod m_i\).

Properties

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

Example

- A RNS base: \((255, 256, 257)\) with \(M = 16776960\).
- \(a = 10000 \rightarrow (55, 16, 234)\) and \(b = 300 \rightarrow (45, 44, 43)\).
- \(a \times b = 3000000 \rightarrow (180, 192, 39)\).
From the Chinese Remainder Theorem

\[ X = \left[ x_1 M_1^{-1} M_1 + x_2 M_2^{-1} M_2 + \ldots + x_n M_n^{-1} M_n \right]_M \] (1)

Where, \( M_i = \frac{M}{m_i} \) and \( M_i^{-1} \) represents the inverse of \( M_i \) modulo \( m_i \).
From the Chinese Remainder Theorem

\[ X = \left| x_1 \left\lfloor M_1^{-1} M_1 \right\rfloor_{m_1} + x_2 \left\lfloor M_2^{-1} M_2 \right\rfloor_{m_2} + \ldots + x_n \left\lfloor M_n^{-1} M_n \right\rfloor_{m_n} \right\rfloor_M \] (1)

Where, \( M_i = \frac{M}{m_i} \) and \( \left\lfloor M_i^{-1} \right\rfloor_{m_i} \) represents the inverse of \( M_i \) modulo \( m_i \).

Find \( \alpha \)

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

- Shenoy Kumaresan, in 1989, proposed to use an extra modulo
- Posch and Posch, in 1995, proposed a floating point approach
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
\[
0 \leq x_i' < m_i
\]
\[
X = x_1' + x_2' m_1 + x_3' m_1 m_2 + \ldots + x_n' m_1 \ldots m_{n-1} \tag{3}
\]

**Parallel algorithm**

\[
\begin{align*}
x_1' &= x_1 \mod m_1 \\
x_2' &= (x_2 - x_1') m_{1,2}^{-1} \mod m_2 \\
x_3' &= ((x_3 - x_1') m_{1,3}^{-1} - x_2') m_{2,3}^{-1} \mod m_3 \\
&\vdots \\
x_n' &= (\cdots (x_n - x_1') m_{1,n}^{-1} - x_2') m_{2,n}^{-1} - \cdots - x_{n-1}') m_{n-1,n}^{-1} \mod m_n
\end{align*}
\]
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

$$\begin{align*}
x'_1 &= x_1 \mod m_1 \\
x'_2 &= (x_2 - x'_1) \mod m_2 \\
x'_3 &= ((x_3 - x'_1) - x'_2 m_1) \mod m_3 \\
&\vdots \\
x'_n &= ((x_n - x'_1) - m_1(x'_2 - m_2(x'_3 - \cdots - m_{n-3}(x'_{n-2} - m_{n-2}x'_{n-1})))) & \ldots
\end{align*}$$
Criteria for conversion

CRT

- Find $\alpha$
- Use $M_i$ and $|M_i|^{-1}_{m_j}$ which are difficult to characterize.
Criteria for conversion

**CRT**
- Find $\alpha$
- Use $M_i$ and $|M_i|^{-1}_{m_j}$ 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

\[ X \mod \tilde{m}_j = [x'_1 + (m_1 - \tilde{m}_j)(x'_2 + (m_2 - \tilde{m}_j)(x'_3 + \ldots + (m_{n-1} - \tilde{m}_j)x'_n)\ldots)] \mod \tilde{m}_j \]

(6)
Remark about MRS to new RNS

From MRS to new RNS: using Horner

\[ X \mod \tilde{m}_j = [x'_1 + (m_1 - \tilde{m}_j)(x'_2 + (m_2 - \tilde{m}_j)(x'_3 + \ldots + (m_{n-1} - \tilde{m}_j)x'_n) \ldots)] \mod \tilde{m}_j \]  

(6)

Proposition for a basis: Minimization of those differences

<table>
<thead>
<tr>
<th>( \hat{d} )</th>
<th>2^8</th>
<th>2^{10}</th>
<th>2^8</th>
<th>2^{10}</th>
<th>2^8</th>
<th>2^{10}</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m )</td>
<td>2^{32}</td>
<td>2^{32}</td>
<td>2^{64}</td>
<td>2^{64}</td>
<td>2^{128}</td>
<td>2^{128}</td>
</tr>
<tr>
<td>( \frac{\hat{d}}{\log(\hat{d})} )</td>
<td>46</td>
<td>147</td>
<td>46</td>
<td>147</td>
<td>46</td>
<td>147</td>
</tr>
<tr>
<td># coprimes found</td>
<td>39</td>
<td>117</td>
<td>44</td>
<td>124</td>
<td>41</td>
<td>121</td>
</tr>
</tbody>
</table>

Table: coprimes found between \( m \) and \( m + d \) with a trivial algorithm
Criteria for conversion

**MRS sequential**
- \( \frac{n^2}{2} \) products by small numbers
- \( n \) classic modular divisions

**MRS parallelize**
- \( \frac{n^2}{2} \) modular divisions by small numbers
Modular Division

Idea

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

Idea

- We want to compute $y = x \times |d|^{-1}_m \mod m$
- 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. $l_m = (-m)^{-1} \mod d$
3. Algorithm:
   1. $x = r_x + q_x \times d$ with $0 \leq r_x < d$
   2. $k \leftarrow r_x \times l_x \mod d$
   3. $y \leftarrow (r_x + k \times r_m)/d + q_x + k \times q_m$  \((\sim y = \frac{x - km}{d})\)
4. Output: $y = x \times d^{-1} \mod m$
Example

### Classic Modular Division

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

Classic Modular Division

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

Modular Division

1. Input: \( m = 10007, d = 15, x = 7856 \)
2. Precompute: \( r_m = 2, q_m = 667, I_m = 7 \)
3. Algorithm:
   1. \( x = r_x + q_x \times d \rightarrow r_x = 11, q_x = 523 \)
   2. \( k \leftarrow 11 \times 7 \mod 15 = 2 \)
   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: $(d^{-1} \mod M) = 4670$
3. Algorithm:
   \[ s = x \times (d^{-1}) \mod M \]
   - $s = 7856 \times 4670 = 36687520$
   - $s = 36687520 \mod 10007 = 1858$

Modular Division

1. Input: $m = 10007$, $d = 15$, $x = 7856$
2. Precompute: $r_m = 2$, $q_m = 667$, $I_m = 7$
3. Algorithm:
   1. $x = r_x + q_x \times d \rightarrow r_x = 11$, $q_x = 523$
   2. $k \leftarrow 11 \times 7 \mod 15 = 2$
   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
     - Splitting: $R \rightarrow R_1, R_0$
     - Call: $(r_{R_1}, q_{R_1}) \leftarrow Barrett_k(R_1, d)$
     - Updating: $R, Q, T$
   - WHILE $R \geq d$ DO $R \leftarrow R - d$, $Q \leftarrow Q + 1$
3. Output: $r_x \leftarrow R$, $q_x \leftarrow Q$
\( \text{Barrett}_k \)

**Generalized Barrett Property**

\[
\begin{align*}
    x - d \left\lfloor \left( \frac{2^{k+\delta}}{d} \left\lfloor \frac{x}{2^\delta - 1} \right\rfloor \right) \frac{2^{k+1}}{2^k + 1} \right\rfloor < 3d
\end{align*}
\]  

(7)
**Barrett**

**Barrett**

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:**
   - \(q \leftarrow \left( \left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor \left\lfloor \frac{x}{2^{\delta-1}} \right\rfloor \right)\)
   - \(q \leftarrow \left\lfloor \frac{q}{2^{k+1}} \right\rfloor\)
   - \(r \leftarrow x - qd \mod 2^{\delta+1}\) (as, \(r < 3d\))
4. **Output:** \((q, r)\) such that \(r = x - dq < 3d\)

**Generalized Barrett Property**

\[
\begin{aligned}
    x - d &\left\lfloor \left( \left\lfloor \frac{2^{k+\delta}}{d} \right\rfloor \left\lfloor \frac{x}{2^{\delta-1}} \right\rfloor \right) \right\rfloor \frac{2^{\delta+1}+1}{2^{k+1}} \end{aligned}
\]

\[
< 3d \quad (7)
\]
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

<table>
<thead>
<tr>
<th></th>
<th>$2^8$</th>
<th>$2^{10}$</th>
<th>$2^8$</th>
<th>$2^{10}$</th>
<th>$2^8$</th>
<th>$2^{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d$</td>
<td>$2^{32}$</td>
<td>$2^{32}$</td>
<td>$2^{64}$</td>
<td>$2^{64}$</td>
<td>$2^{128}$</td>
<td>$2^{128}$</td>
</tr>
<tr>
<td>$m$</td>
<td>Our approach</td>
<td>1252</td>
<td>1521</td>
<td>2386</td>
<td>2868</td>
<td>4219</td>
</tr>
<tr>
<td></td>
<td>Barrett algo</td>
<td>2176</td>
<td>2176</td>
<td>8448</td>
<td>8448</td>
<td>33280</td>
</tr>
</tbody>
</table>

**Table:** Number of binary operations for a modular division
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.