ADVANCED ARCHITECTURES

RESIDUE NUMBER SYSTEM

Annalisa Massini

Lectures 6-7

2024-2025

RESIDUE NUMBER SYSTEM

Computer Arithmetic Algorithms – I. Koren – 2nd Ed

Ch. 11 The Residue Number System

Computer Arithmetic – Algorithms and Hardware Designs – B. Parhami – 2nd Ed

Ch. 4 Residue Number Systems

Residue number systems

- The **residue number system** is an integer number system whose most important property is that **additions**, **subtractions**, and **multiplications** are inherently **carry-free**
- Unfortunately, **other operations** like division comparison and sign detection are **complex** and **slow**
- Also, residue number systems are **not convenient** when we want to represent **fractions**
- For **special-purpose applications**, such as many types of digital filters, in which the number of additions and multiplications is much greater than invocations of magnitude comparison, overflow detection, division and alike, the residue number system can be very **actractive**

Residue number systems

- **Residue number systems** are based on the *congruence* relation:
  - Two integers *a* and *b* are said to be ***congruent modulo m*** if *m* divides exactly the difference of *a* and *b*
  - We write $a \equiv b \pmod{m}$
- For example
  - $10 \equiv 7 \pmod{3}$
  - $10 \equiv 4 \pmod{3}$
  - $10 \equiv 1 \pmod{3}$
  - $10 \equiv -2 \pmod{3}$
- The number ***m*** is a ***modulus*** or *base*, and we assume that its values exclude 1, which produces only trivial congruences
- A **residue number system** is characterized by a base that is not a single radix but a tuple of integers

Residue number systems

- If ***q*** and ***r*** are the **quotient** and **remainder**, respectively, of the integer division of a by m, that is: $a = qm + r$

  then, by definition, we have $a \equiv r \pmod{m}$

- The number ***r*** is said to be the ***residue*** of ***a*** with respect to ***m***, and we shall usually denote this by $r = |a|_m$

- The set of m values $\{0; 1; 2; \ldots ; m - 1\}$ that the residue may assume is called the set of *least positive residues modulo m*

Residue number systems

- Suppose we have a set $\{m_1; m_2; \ldots; m_N\}$ of N positive and pairwise **relatively prime** moduli

- Let M be the **product of the moduli** $M = m_1 x m_2 x \ldots x m_N$
- M is the ***dynamic range*** and $[0; M\text{-}1]$ is the range of representation

- We write the **representation of *X*** in the form **$< x_1; x_2; \ldots; x_N >$** (or **$< x_N; x_{N-1}; \ldots; x_1 >$),** where **$x_i = |X|_{mi}$** , and we indicate the relationship between X and its residues by writing

$$X \approx < x_1; x_2; \ldots; x_N >$$

- **Example** Consider the residue system {2, 3, 5}, then M=30

8 is represented as < 0, 2, 3 >

16 is represented as < 0, 1, 1 >

Residue number systems

- Every number ***X < M*** has a **unique representation** in the residue number system, which is the sequence of residues $<|X|_{m_i} \mid 1 \le i \le N>$
- A (partial) proof of **uniqueness** is as follows:
  - Suppose X_1 and X_2 are two different numbers with the **same *residue representation***
  - Then $|X_1|_{m_i} = |X_2|_{m_i}$, and so $|X_1 - X_2|_{m_i} = 0$
  - Therefore $X_1 - X_2$ is the least common multiple (**lcm**) of m_i
  - But if the m_i are **relatively prime**, then their **lcm** is M, and it must be that $X_1 - X_2$ is a multiple of M
  - So it cannot be that $X_1 < M$ and $X_2 < M$
  - Therefore, the representation $<|X|_{m_i} : 1 \le i \le N>$ is unique and may be taken as the representation of X

Residue number systems

- Representations in a system in which the moduli are **not pairwise relatively prime** will be **not unique**: two or more numbers will have the same

- For a given M we can represent:
  - **Nonnegative integers** only
  - Or intervals with also **negative integers** giving a rule to represent them

	Relatively prime			Relatively non-prime		
N	m1=2	m2=3	m3=5	m1=2	m2=4	m3=6
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	0	2	2	0	2	2
3	1	0	3	1	3	3
4	0	1	4	0	0	4
5	1	2	0	1	1	5
6	0	0	1	0	2	0
7	1	1	2	1	3	1
8	0	2	3	0	0	2
9	1	0	4	1	1	3
10	0	1	0	0	2	4
11	1	2	1	1	3	5
12	0	0	2	0	0	0
13	1	1	3	1	1	1
14	0	2	4	0	2	2
15	1	0	0	1	3	3

Residue number systems

For a given M:

- If only **nonnegative integers** are needed, the range is [0, M-1]
- If **negative numbers** are also desired, the range can be set to:
  - [-(M-1)/2,(M-1)/2] if M is odd
  - [-M/2,(M/2)-1] if M is even

	m1=2	m2=3	m3=5	
0	0	0	0	
1	1	1	1	
2	0	2	2	
3	1	0	3	
4	0	1	4	
5	1	2	0	
…	….	….	….	
14	0	2	4	
15	1	0	0	
16	0	1	1	-14
17	1	2	2	-13
18	0	0	3	-12
…	….	….	….	…

Residue number systems

- The definition of **negative values** is given using the concept of **additive inverse**
- The **additive inverse** is obtained by complementing the residues with respect to its modulus

$$\langle X\rangle_{m_i} = \begin{cases} \langle X\rangle_{m_i} & \text{if } X \geq 0 \\ \left\langle m_i - \langle |X| \rangle_{m_i} \right\rangle_{m_i} & \text{if } X < 0 \end{cases}$$

	m1=2	m2=3	m3=5	
0	0	0	0	
1	1	1	1	
2	0	2	2	
3	1	0	3	
4	0	1	4	
5	1	2	0	
…	….	….	….	
14	0	2	4	
15	1	0	0	
16	0	1	1	-14
17	1	2	2	-13
18	0	0	3	-12
…	….	….	….	…

RNS addition and multiplication

- Residue **addition** and **multiplication** are carried out by individually adding/multiplying the corresponding digits
- A **carry-out** from one digit position is **not propagated** into the next digit position

- **Example** Consider the moduli-set {2*;* 3*;* 5} where M=30
  - Operand A 11 <1; 2; 1>
  - Operand B 2 <0; 2; 2>
  - Sum 13 <1; 1; 3>
  - Product 22 <0; 1; 2>

- Note that it is difficult to see that 22 > 13 by looking at the representation of the values

RESIDUE NUMBER SYSTEM

Selecting the moduli

Selecting the moduli

- When selecting the moduli we can follow **different objectives**
- To reduce the execution time of additions and multiplications, then a large number of small moduli is desirable
  - the execution time is determined by the largest module
- On the other hand, a large number of small modules
  - Requires more units, one for each module
  - Increases the conversion time, since a sequential procedure whose number of steps depend on the number of modules is requires
- Also, residues will be coded in binary and arithmetic operations will be executed on the corresponding binary representations
- In summary, the objectives when selecting the RNS modules are:
  - **Representational efficiency**
  - **Complexity of arithmetical operations**

Selecting the moduli

- Let us consider an **example**
  - Represent unsigned integers in the range [0, 100000)
  - 17 bits are required with unsigned binary representation
- A simple strategy is to pick **prime numbers** in sequence until the dynamic range M is obtained

- If we pick $m_0 = 2$, $m_1 = 3$, …, $m_5 = 13$

  $< 13, 11, 7, 5, 3, 2 > \rightarrow M=30030$

- This is not enough, hence we add also $m_6 = 17$

  $< 17, 13, 11, 7, 5, 3, 2 > \rightarrow M=510510$

- Now the dynamic range is more than 5 times larger than needed

Selecting the moduli

- If we remove $m_2 = 5$ we get

  < 17, 13, 11, 7, 3, 2 > → M=102102

- The binary encoding of the six residues requires

  5 + 4 + 4 + 3 + 2 + 1 = 19 bits

- The speed of arithmetic operations is dictated by the 5-bit residue → to balance, we can combine the pairs of moduli 2 and 13, and 3 and 7, with no speed penalty
- This leads to:

  < 26, 21, 17, 11 > → M=102102

- This alternative RNS still needs 5+5+5+4 = 19 bits per operand, but has two **fewer modules** in the arithmetic unit

Selecting the moduli

- Better results can be obtained if we include **powers of smaller primes** before moving to larger primes
- Note that powers of two prime numbers are relatively prime
- For example, instead of including $m_0 = 2$ and $m_1 = 3$, we can take $\mathbf{m_0 = 3}$ **and** $\mathbf{m_1 = 2^2}$ (that are smaller than the next prime 5)
- Similarly, after $m_2 = 5$ and $m_3 = 7$, we can take $\mathbf{2^3}$ **and** $\mathbf{3^2}$ (that are smaller than the next prime 11)
- And we have

$$< 13, 11, 3^2, 2^3, 7, 5 > \rightarrow \qquad M=360360$$

Selecting the moduli

- Since the dynamic range is too large (3.6 times), we can replace the 9 with 3 and then combine the pair 5 and 3 to obtain

$$< 15, 13, 11, 2^3, 7 > \quad \rightarrow \quad M=120120$$

- The **binary encoding** of the residues requires

$$4 + 4 + 4 + 3 + 3 = 18 \text{ bits}$$

- This is better than earlier result of 19 bits

- And the **speed has also improved** because the largest residue is now 4 bits wide instead of 5

Selecting the moduli

- It is also important to have moduli sets that, besides an efficient representation, facilitate the balance, such that the ***differences between the moduli is as small as possible***

- Consider, for example, the choice of **13 and 17** for the moduli: they are adjacent prime numbers and give **M = 221**

- The binary encoding requires 4 + 5 = 9 bits
- The representational efficiency is:
  - In the first case 13/16
  - In the second case is 17/32

Selecting the moduli

- If we choose 13 and 16, then:
  - the representational efficiency is **improved** to 16/16 in the second case
  - but we have a **reduction in the range** down to 208

- With the **better balanced pair** 15 and 16, we will have:
  - A better representational efficiency, that is 15/16 and 16/16 respectively
  - A greater dynamic range M=240

- Note that in this case the moduli are related to **a power of 2**, that is $15 = 2^4-1$ and $16 = 2^4$

- In general, to maximize the representational efficiency, modules m_i equal $\mathbf{2^k}$ or very close to it, as $\mathbf{2^k-1}$, are preferred

Selecting the moduli

- Choosing modules that are in the form **2^k** or **2^k-1** implies that arithmetic on residue digits do ***not deviate too far from conventional arithmetic***, which is just **arithmetic modulo a power of 2**
- Modules in the form **2^k** and **2^k-1** are called **low-cost** modules
- Clearly, we can select **only one m_i** in the form **2^k**
- Then we can select **$2^k – 1$** and a few other modules in the form **$2^l – 1$**
- **But** not all pairs of numbers of the form $2^i – 1$ are relatively prime
- It can be shown that that 2^j - 1 and 2^k - 1 are relatively prime if and only if ***j*** and ***k*** are **relatively prime**

Selecting the moduli

- For example:
  - 2^4-1= 15 — 15=3x5
  - 2^5-1= 31 — ***31 prime***
  - 2^6-1= 63 — 63=3x7
  - 2^7-1= 127 — ***127 prime***
  - 2^8-1= 255 — 255=3x5x17

- **Example** Consider RNS **< 2^5, 2^5 -1, 2^4-1, 2^3-1 > = < 32, 31, 15, 7 >**
  - The total number of bit required is: 5 + 5 + 4 + 3 = 17 bits
  - M = 104160 = 2^5 x (2^5 -1) x (2^4-1) x (2^3-1) > 2^{16}
  - The representational efficiency is close to 100% and no bit is wasted

- In general, a representational efficiency better than 50% leads to the waste of no more than 1 bit in number representation

Selecting the moduli

- Many moduli sets are based on these choices, but there are other possibilities
- For example, moduli-sets of the form $\{2^n-1; 2^n; 2^n+1\}$ are among the most popular in use

- In summary, four considerations for the selection of moduli
  - The selected moduli must provide an **adequate range** whilst also ensuring that RNS representations are **unique**
  - The **efficiency of binary representations** as well as the balance between the different moduli in a given moduli-set are also important
  - The **implementations of arithmetic units** for RNS should be compatible with those for conventional arithmetic
  - The **size of individual moduli**

RESIDUE NUMBER SYSTEM

RNS operations

RNS arithmetic operations

- One of the primary **advantages** of RNS is that RNS-arithmetic operation like **addition, subtraction and multiplication do not require carries** between digits

- But, this is true only between *digits*

- Since a digit is ultimately represented in binary, there will be carries between bits, and therefore it is important to ensure that digits (that is the moduli) are **not too large**

- Notice also that **small digits** make it possible to realize cost-effective **table-lookup implementations** of arithmetic operations

RNS arithmetic operations

Basic arithmetic

- Addition/subtraction and multiplication are easily implemented with residue notation, depending on the choice of the moduli
- Division is much more difficult due to the difficulties of sign-determination and magnitude-comparison

Negative numbers

- As with the conventional number systems, any one of the radix complement, diminished-radix complement, or sign-and-magnitude notations may be used in RNS
- The merits and drawbacks of choosing one over the other are similar to those for the conventional notations

RNS addition

- Residue **addition** is carried out by individually adding corresponding digits
- A **carry-out** from one digit position is **not propagated** into the next digit position
- **Example** Consider the moduli-set {2; 3; 5; 7} where M=210

• Operand 1	17	<1; 2; 2; 3>
• Operand 2	19	<1; 1; 4; 5>
• Result	36	<0; 0; 1; 1>

RNS subtraction

- **Subtraction** may be carried out by obtaining the additive inverse of the subtrahend (in the chosen notation) and adding to the minuend
- **Example** Consider the moduli-set {2; 3; 5; 7} where M=210
  - 17 <1; 2; 2; 3>   19 <1; 1; 4; 5>
  - -17 <1; 1; 3; 4>   -19 <1; 2; 1; 2>

Subtraction		**19 -17**		**17-19**
	19	<1; 1; 4; 5>	17	<1; 2; 2; 3>
	-17	<1; 1; 3; 4>	-19	<1; 2; 1; 2>
	2	<0; 2; 2; 2>	-2	<0; 1; 3; 5>

- Easy for numbers in diminished-radix complement or radix complement notation
- More expensive for sign-and-magnitude representation, where a slight modification of the algorithm is necessary

RNS addition and subtraction

- For moduli in the form m = $\mathbf{2^k}$ an ordinary binary adder can be used, and the additive inverse is the 2's complement
- For moduli in the form m = $\mathbf{2^k - 1}$ we need an adder with end-around carry, and the additive inverse is the 1's complement, that is $m - c = 2^k - 1 - c$

Example

- Let us consider $l = 3$ and $m = 2^l - 1 = 7$
- To execute 6 - 4, we add the 1's complement of 4 to 6 with end-around carry

$$\begin{array}{rr l}
 & 110 & 6 \\
+ & 011 & 1\text{'s complement of } 100 \text{ of } 4 \\
\hline
1 & 001 & \\
+ & 1 & \text{end} - \text{around carry} \\
\hline
 & 010 &
\end{array}$$

RNS multiplication

Basic arithmetic -

- **Multiplication** too can be performed simply by multiplying corresponding residue digit-pairs, relative to the modulus for their position → multiply digits and ignore or adjust an appropriate part of the result

- **Example** Consider the moduli set {2; 3; 5; 7}
  - Operand 1 11 <1; 2; 1; 4>
  - Operand 2 13 <1; 1; 3; 6>
  - Product 323 <1; 2; 3; 3>

RNS division

- Basic fixed-point **division** consists of a sequence of subtractions, magnitude-comparisons, and selections of the quotient-digits
- Anyway, **comparison** in RNS is a difficult operation, because RNS is not positional or weighted

- **Example** Consider the moduli set {2; 3; 5; 7}:
  - the number represented by <0; 0; 1; 1> = 36 is almost twice that represented by <1; 1; 4; 5> = 19
  - but this is far from apparent

RESIDUE NUMBER SYSTEM

RNS associated mixed-radix system

The associated mixed-radix system

- Magnitude comparison, sign detection and overflow detection can be facilitated by converting the given residue representations into the **associated mixed-radix system**

- This is a **weighted number system**, where the representation for a number *Y* is:

$$Y = z_N \cdot (m_{N-1} \cdot m_{N-2} \cdots m_1) + \cdots + z_3 \cdot (m_1 \cdot m_2) + z_2 \cdot m_1 + z_1$$

  with digits z_i satisfying $0 \leq z_i \leq m_i$, that is the same range as RNS digits

- **Example** Consider the moduli set {8; 7; 5; 3}
  - $(0 \mid 3 \mid 1 \mid 0)_{MRS(8|7|5|3)} = 0\times105 + 3\times15 + 1\times3 + 0\times1 = 48$

The associated mixed-radix system

- RNS-to-MRS **conversion problem** is determining z_i digits of MRS given the x_i digits of RNS:

  $X = \langle x_{k-1} ; \; ... \; ; x_2 ; x_1 ; x_0 \rangle_{RNS} = (z_{k-1} \mid ... \mid z_2 \mid z_1 \mid z_0)_{MRS}$

Example Consider **48 and 45** in the moduli set {8; 7; 5; 3}

$48 = \langle 0 ; 6 ; 3 ; 0 \rangle_{RNS}$

$\langle 000 ; 110 ; 011 ; 00 \rangle_{RNS}$

$45 = \langle 5 , 3 ; 0 ; 0 \rangle_{RNS}$

$\langle 101 ; 011 ; 000 ; 00 \rangle_{RNS}$

Equivalent **mixed-radix representations**

$(0 \mid 3 \mid 1 \mid 0)_{MRS}$

$(000 \mid 011 \mid 001 \mid 00)_{MRS}$

$(0 \mid 3 \mid 0 \mid 0)_{MRS}$

$(000 \mid 011 \mid 000 \mid 00)_{MRS}$

- With MRS the magnitude comparison is straightforward

The associated mixed-radix system

- **RNS-to-MRS conversion** From the definition of

$$Y = z_N \cdot (m_{N-1} \cdot m_{N-2} \cdots m_1) + \cdots + z_3 \cdot (m_1 \cdot m_2) + z_2 \cdot m_1 + z_1$$

- Immediately follows that $\boldsymbol{z_1 = x_1}$
- To obtain the value of $\boldsymbol{z_2}$:
  - First we subtract $z_1 = x_1$ from both the RNS and MRS representations
  - Then divide both representations by m_1,
  - We get the expression of z_2 using x_2 and x_1
- Repeating the same process we can determine **all the z_i**
- Division $Y' = Y - x_1$ by m_1 (operation known as *scaling*) can be executed by multiplying each residue by the multiplicative inverse of m_1 with respect to the associated modulus
- **Example** The multiplicative inverses of 3 relative to 8, 7, and 5 are 3, 5, and 2, respectively, because

$$(3 \times 3)_8 = (3 \times 5)_7 = (3 \times 2)_5 = 1$$

Residue number systems

Forward conversion

- The most direct way to convert from a conventional representation to a residue one is to divide by each of the given moduli and then collect the remainders
- This is a **costly** operation if the number is represented in an **arbitrary radix** and the **moduli are arbitrary**
- If number is represented in **radix-2** (or a radix that is a power of two) and the moduli are of a suitable form (e.g. 2^n-1), then these procedures that can be implemented with more efficiency

Residue number systems

Reverse conversion

- The conversion from residue notation to a conventional notation is more difficult and so far has been one of the major impediments to the adoption use of RNS
  - One method is to first derive the **mixed-radix** representation of the RNS number and then use the weights of the mixed-radix positions to complete the conversion
  - We can also derive position weights for the RNS directly based on the **Chinese remainder theorem** (CRT)
    - The Chinese remainder theorem

$$X = \langle x_N;\ \cdots;\ x_1 \rangle = \left\langle \sum_{i=1}^{N} M_i \langle \alpha_i x_i \rangle_{m_i} \right\rangle_M$$

where, by definition, $M_i = M/m_i$ and $\alpha_i = \langle M_i^{-1} \rangle_{m_i}$ is the multiplicative inverse of M_i with respect to m_i

Residue number systems

Exercise

Consider the RNS system < 13 ; 11 ; 8 ; 7 >

a. Represent the numbers $x = 68$ and $y = 23$

b. Compute $x + y$, $x - y$, $x \times y$, checking the results

c. Represent $x = 68$ using mixed-radix system

d. Compute the representational efficiency of this RNS compared with standard binary