

Examples of Standard and Exotic Number Systems

Arnaud Tisserand

CNRS, IRISA laboratory, CAIRN research team

Séminaire au vert CAIRN  
Erdeven, Dec. 1-3, 2010



- integer and fixed point
- floating point (IEEE standards and revision)
- (semi-)logarithmic number system ((S)LNS)
- basic redundant number systems
- double base number system (DBNS)
- residue number system (RNS)

Babylonian Arithmetic

Use of a **positional number system** with:

- primary **radix 60**
- auxiliary radix 10
- **digits** in the set:

1	2	3	4	5	6	7	8	9	10

Example:

=  $33 \times 60 + 30 = 2010$

Almost New Type of Look-Up Tables

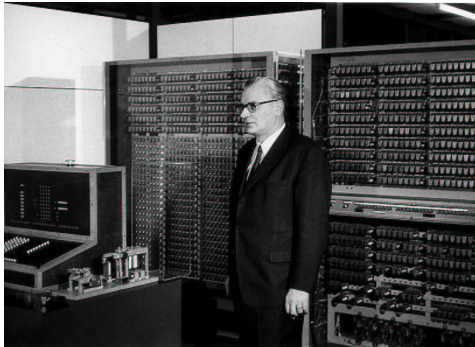

Illustration of a:

- multiplication by 25 table
- discovered in Susa
- dated to approx. -2000
- preserved in Louvre museum

Remark: only the products by (1), 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 30, 40, 50 are required (not all the 59 products)

## First Computer with a Floating-Point Unit

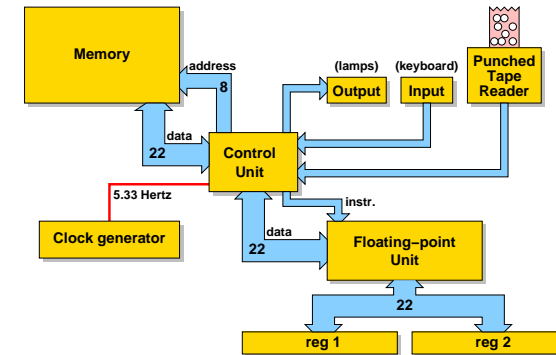
Z3 designed by Konrad Zuse (1910–1995) in 1941, Berlin



picture of the version rebuilt in 1961

Source: <http://www.epemag.com/zuse/>

## Z3: Architecture and Characteristics



size	5 m × 2 m × 0.8 m
weight	≈ 1000 kg
frequency	5.33 Hz
technology	elect. relays (num.: 600, mem.: 1400)
power consumption	≈ 4000 W

## Z3 : Data Format and Floating-Point Unit

Floating-point representation:

- 1-bit sign
- 7-bit exponent represented using 2's complement
  - ▶ exponent = -64 → 0
  - ▶ exponent = 63 → ∞
- 14-bit mantissa + 1 implicit bit

Floating-point unit:

- addition/subtraction 3 cycles (0.6s/op.)
- multiplication 16 cycles (3.0s/op.)
- division 18 cycles (3.4s/op.)
- square root (variable latency)

Other instructions: read keyboard, display, load, store...

## Number Systems

- set of represented numbers
  - ▶ integers:  $\mathbb{N}, \mathbb{Z}$
  - ▶ rationals:  $\mathbb{Q}$
  - ▶ real approximations: subsets of  $\mathbb{R}$
  - ▶ complex approximations: subsets of  $\mathbb{C}$
  - ▶ finite fields:  $\mathbb{F}_p, \mathbb{F}_{2^m}, \mathbb{F}_{3^m}$
  - ▶ ...
- system properties
  - ▶ positional or non positional
  - ▶ redundant or non redundant
  - ▶ fixed precision or arbitrary precision (multiple precision)
  - ▶ completeness (in a finite set)
  - ▶ ...

Number system =

1. data format and encoding
2. a set of interpretation rules for the encoding

## Positional Number System(s)

$$X = \sum_{i=-m}^{n-1} x_i \beta^i = (x_{n-1}x_{n-2} \cdots x_1x_0 \cdot x_{-1}x_{-2} \cdots x_{-m})$$

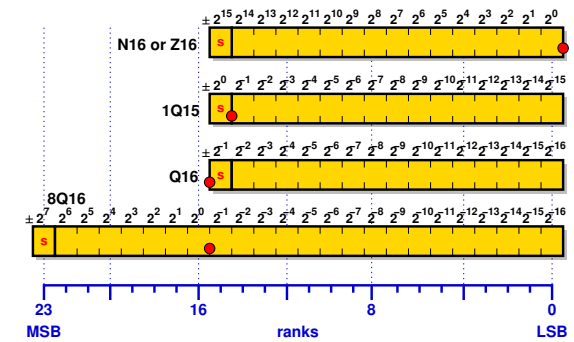
- **radix**  $\beta$  (usually a power of 2)
- **digits**  $x_i$  ( $\in \mathbb{N}$ ) in the **digit set**  $\mathcal{D}$
- **rank** or **position**  $i$ , **weight**  $\beta^i$
- $n$  **integer** digits,  $m$  **fractional** digits

Examples:

- $\beta = 10, \mathcal{D} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- $\beta = 2, \mathcal{D} = \{0, 1\}$
- carry save:  $\beta = 2, \mathcal{D}_{cs} = \{0, 1, 2\}$
- borrow save:  $\beta = 2, \mathcal{D}_{bs} = \{-1, 0, 1\}$
- signed digits:  $\beta > 2, \mathcal{D}_{sd,\alpha,\beta} = \{-\alpha, \dots, \alpha\}$  with  $2\alpha + 1 \geq \beta$
- theoretical systems:  $\beta = \frac{1+\sqrt{5}}{2}, \beta = 1 + i \dots$

## Fixed-Point Representations

Widely used in DSPs and digital integrated circuits for higher speed, lower silicon area and power consumption compared to floating point



Typical fixed-point formats: 16, 24, 32 and 48 bits

## Floating-Point Representation(s)

Radix- $\beta$  floating-point representation of  $x$ :

- **sign**  $s_x$ , 1-bit encoding:  $0 \Rightarrow x > 0$  and  $1 \Rightarrow x < 0$
- **exponent**  $e_x \in \mathbb{N}$  on  $k$  digits and  $e_{min} \leq e_x \leq e_{max}$
- **mantissa**  $m_x$  on  $n + 1$  digits
- encoding:

$$x = (-1)^{s_x} \times m_x \times \beta^{e_x}$$

$$m_x = x_0 \cdot x_1 x_2 x_3 \cdots x_n$$

$$x_i \in \{0, 1, \dots, \beta - 1\}$$

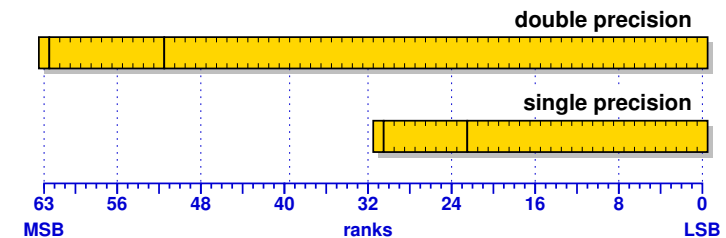
For accuracy purpose, the mantissa must be **normalized** ( $x_0 \neq 0$ )

Then  $m_x \in [1, \beta[$  and a specific encoding is required for the number 0

## IEEE-754: basic formats

Radix  $\beta = 2$ , the first bit of the normalized mantissa is always a "1" (non-stored implicit bit)

format	number of bits			
	total	sign	exponent	mantissa
double precision	64	1	11	52 + 1
single precision	32	1	8	23 + 1



## IEEE-754: Exponent and Special Values

format	size $k$	bias $b$	unbiased		biased	
			$e_{min}$	$e_{max}$	$e_{min}$	$e_{max}$
SP	8	127 ( $= 2^{8-1} - 1$ )	-126	127	1	254
DP	11	1023 ( $= 2^{11-1} - 1$ )	-1022	1023	1	2046

-0	1	00000000	000000000000000000000000
+0	0	00000000	000000000000000000000000
$-\infty$	1	11111111	000000000000000000000000
$+\infty$	0	11111111	000000000000000000000000
NaN	0	11111111	000000000000000000000001 (for instance)

Not a Number (NaN) is the result of **invalid operations** such as  $0/0$ ,  $\sqrt{-1}$  or  $0 \times \infty$

## Floating-Point Standard Revision: IEEE 754-2008

Standards and revisions:

- 754-1985 binary
- 854-1987 radix-independent
- 754-2008

Main evolutions in 754-2008:

- radix **2** (binary) and radix **10** (decimal)
- FMA: **fused multiply and add** ( $a \times b \pm c$ )
- support of the **quadruple** precision
- emphasis on *portability*, *reproducibility* and *performances*

## IEEE 754-2008: Formats

format	total [b]	mantissa [d]	exponent [b]	exponent range
binary16	16	10+1	5	[-14, 15]
binary32	32	23+1	8	[-126, 127]
binary64	64	52+1	11	[-1022, 1023]
binary128	128	112+1	15	[-16382, 16383]
decimal32	32	7	11	[-95, 96]
decimal64	64	16	13	[-383, 384]
decimal128	128	34	17	[-6143, 6144]

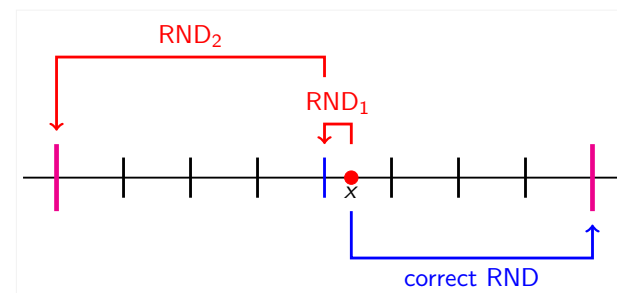
## Double Rounding Problem(s)

Example:  $s = a + b$  on a x86 architecture gives

```

fldl a;      load a
faddl b;     load b and add it to a in extended precision
fstpl s;     store result in DP
    
```

RND<sub>1</sub>  
RND<sub>2</sub>



## Logarithmic Number System (LNS)

Representation of  $x$ :

(zero flag, sign of  $x$ , fixed-point approximation of  $\log_2 x$ )

LNS operations:

$$\log_2(a \times b) = \log_2 a + \log_2 b$$

$$\log_2(a \div b) = \log_2 a - \log_2 b$$

$$\log_2(a \pm b) = \log_2 a + \log_2(1 \pm 2^{\log_2 b - \log_2 a})$$

$$\log_2(a^{\frac{p}{q}}) = \frac{p}{q} \times \log_2 a$$

where the functions  $\log_2(1 + 2^x)$  and  $\log_2(1 - 2^x)$  are approximated (tables or polynomials)

Applications in digital signal processing and digital control

## Semi-Logarithmic Number System (SLNS)

Representation of  $x$ :

- sign:  $s_x = \pm 1$
- exponent:  $e_{k,x}$  is a multiple of  $2^{-k}$
- mantissa:  $1 \leq m_{k,x} < 1 + 2^{-k}$

$$x = s_x \times m_{k,x} \times 2^{e_{k,x}}$$

where

$$e_{k,x} = \underbrace{\text{bbbb} \cdots \text{bbb}}_k \cdot \underbrace{\text{bbb} \cdots \text{bb}}_k$$

$$m_{k,x} = 1. \underbrace{\text{0000} \cdots \text{000}}_k \text{bbb} \cdots \text{bb} = 1 + \varepsilon_x$$

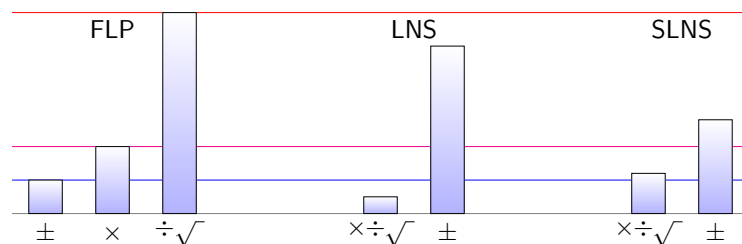
$$m_1 \times m_2 = (1 + \varepsilon_1) \times (1 + \varepsilon_2) \simeq 1 + \varepsilon_1 + \varepsilon_2$$

## FLP vs. LNS vs. SLNS

$$X_{\text{FLP}} = (-1)^b \times 1.\underbrace{\text{bbbbbbbbbbbbbb}}_n \times 2^{\underbrace{\text{bbbbbb}}_e}$$

$$X_{\text{LNS}} = (1 - b) \times (-1)^b \times 2^{\underbrace{\text{bbbbbb}}_e} \cdot \underbrace{\text{bbbbbbbbbbbbbb}}_n$$

$$X_{\text{SLNS}} = (-1)^b \times 1.\underbrace{\text{00000000}}_{\approx n/2} \underbrace{\text{bbbbbb}}_{\approx n/2} \times 2^{\underbrace{\text{bbbbbb}}_e} \cdot \underbrace{\text{bbbbbb}}_{\approx n/2}$$



## Redundant or Constant Time Adders

To speed-up the addition, one solution consists in “saving” the carries and using them (this makes sense only in case of multiple additions)

In 1961, Avizienis suggested to represent numbers in radix  $\beta$  with digits in  $\{-\alpha, -\alpha + 1, \dots, 0, \dots, \alpha - 1, \alpha\}$  instead of  $\{0, 1, 2, \dots, \beta - 1\}$  with  $\alpha \leq \beta - 1$

Using this representation, if  $2\alpha + 1 > \beta$  some numbers have several possible representation at the bit level. For instance, the value 2345 (in the standard representation) can be represented in radix 10 with digits in  $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$  by the values 2345, 235(-5) or 24(-5)(-5)

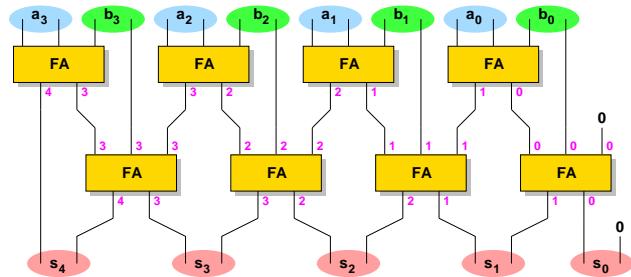
Such a representation is said **redundant**

In a redundant number system there is constant-time addition algorithm (without carry propagation) where all computations are done in parallel

## Carry-Save Adder

In **carry-save**, the number  $A$  is represented in radix 2 using digits  $a_i \in \{0, 1, 2\}$  coded by 2 bits such that  $a_i = a_{i,c} + a_{i,s}$  where  $a_{i,c} \in \{0, 1\}$  and  $a_{i,s} \in \{0, 1\}$

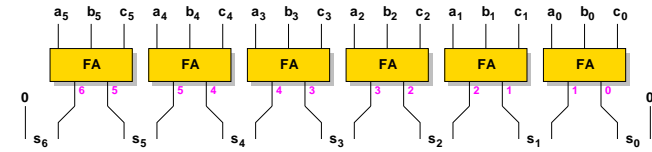
$$A = \sum_{i=0}^{n-1} a_i 2^i = \sum_{i=0}^{n-1} (a_{i,c} + a_{i,s}) 2^i$$



Carry-save addition: delay of 2 FA cells ( $T = O(1)$ )

## Carry-Save Trees

Example with 3 inputs:  $A$ ,  $B$  and  $C$



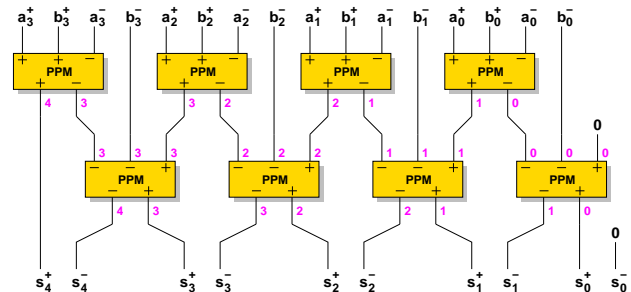
**Carry-save reduction tree**:  $n(h)$  non-redundant inputs can be reduced by a  $h$ -level carry-save tree where  $n(h) = \lfloor 3n(h-1)/2 \rfloor$  and  $n(0) = 2$

$h$	1	2	3	4	5	6	7	8	9	10	11
$n(h)$	3	4	6	9	13	19	28	42	63	94	141

## Borrow-Save Addition

In **borrow-save**, the number  $A$  is represented in radix 2 using digits  $a_i \in \{-1, 0, 1\}$  coded by 2 bits such that  $a_i = a_i^+ - a_i^-$  where  $a_i^+ \in \{0, 1\}$  and  $a_i^- \in \{0, 1\}$

$$A = \sum_{i=0}^{n-1} a_i 2^i = \sum_{i=0}^{n-1} (a_i^+ - a_i^-) 2^i$$



Borrow-save addition: delay of 2 PPM cells ( $T = O(1)$ )

## PPM Cell

Arithmetic equation:

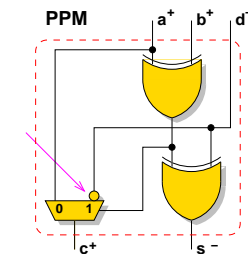
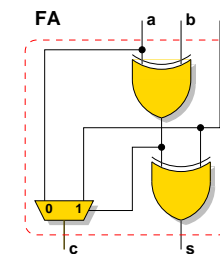
$$2c^+ - s^- = a^+ + b^+ - d^-$$

Logic equation:

$$s = a^+ \oplus b^+ \oplus d^-$$

$$c = a^+ b^+ + a^+ \overline{d^-} + b^+ \overline{d^-}$$

$a^+$	$b^+$	$d^-$	$c^+$	$s^-$
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	0	0
1	0	0	1	1
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1



## ECC: Scalar Multiplication

This is the main operation for ECC

**Inputs:**  $P$  a point of the curve  $E$ , a large integer  $k = \sum_{i=0}^{n-1} k_i 2^i$

**Output:** the point  $Q = [k]P = \underbrace{P + P + P + \dots + P}_{k \text{ times}}$

Basic algorithm: *double-and-add*

- 1:  $Q \leftarrow P$
- 2: **for**  $i$  **from**  $n-2$  **to**  $0$  **do**
- 3:      $Q \leftarrow 2P$
- 4:     **if**  $k_i = 1$  **then**  $Q \leftarrow Q + P$

**Problem:** weak for SPA!

## Countermeasure: Key Recoding

Recoding:  $w$ -NAF (*non-adjacent form*)

With

$$k = \sum_{i=0}^{n-1} k_i 2^i, \quad k_i \in \{0, 1\}$$

use  $k$  with digits in “windows” of  $w$  bits

$$|k_i| < 2^{w-1}$$

Example:

$$k = 267 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}_2$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & \bar{1} & 0 & \bar{1} \end{pmatrix}_{2-NAF}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 3 \end{pmatrix}_{3-NAF}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & \bar{5} \end{pmatrix}_{4-NAF}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 \end{pmatrix}_{5-NAF}$$

Cost:  $n - 1$  DBL and  $\frac{n}{w+1}$  ADD

## Double-Base Number Systems (DBNS) (1/3)

**Source:** L. Imbert

**Redundant** representation based the sum of powers of 2 **AND** 3:

$$x = \sum_{i=1}^n x_i 2^{a_i} 3^{b_i}, \quad \text{with } x_i \in \{-1, 1\}, \quad a_i, b_i \geq 0$$

Example:  $127 = 108 + 16 + 3 = 72 + 54 + 1 = \dots$

	1	2	4	8	16
1					1
3	1				
9					
27			1		

	1	2	4	8
1	1			
3				
9				1
27		1		

## Double-Base Number Systems (DBNS) (2/3)

Smallest  $x > 0$  with  $n$  DBNS terms in its decomposition:

$n$	unsigned	signed
2	5	5
3	23	105
4	431	(4985)
5	18,431	?
6	3,448,733	
7	1,441,896,119	
8	?	

DBNS is a very **sparse** and **redundant** representation

Example: 127 has 783 DBNS representations among which 6 are canonic:  $127 = (108 + 18 + 1) = (108 + 16 + 3) = (96 + 27 + 4) = (72 + 54 + 1) = (64 + 54 + 9) = (64 + 36 + 27)$

## Double-Base Number Systems (DBNS) (3/3)

Application: ECC scalar multiplication

$$314159 = 2^4 3^9 + 2^8 3^1 - 1$$

$$[314159]P = [2^4 3^9]P + [2^8 3^1]P - P$$

cost: 12 DBL + 10 TPL + 2 ADD

$$314159 = 2^4 3^9 - 2^0 3^6 - 3^3 - 3^2 - 3 - 1$$

$$[314159]P = 3(3(3(3^3([2^4 3^3]P - P) - P) - P) - P) - P$$

cost: 4 DBL + 9 TPL + 5 ADD

## Residue Number System (RNS)

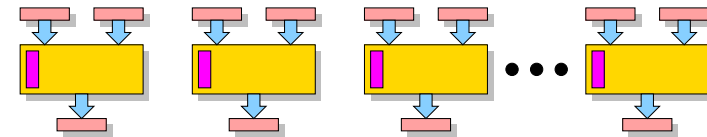
- Base  $\mathcal{B} = (m_1, m_2, \dots, m_k)$  of  $k$  relatively prime moduli
- Size of the base:  $k$

$$A = \{a_1, a_2, \dots, a_k\}, \quad \forall i \ a_i = A \bmod m_i$$

Operations:

$$A \pm B = (|a_1 \pm b_1|_{m_1}, \dots, |a_k \pm b_k|_{m_k})$$

$$A \times B = (|a_1 \times b_1|_{m_1}, \dots, |a_k \times b_k|_{m_k})$$



## Residue Number System: Example (1/2)

Base:  $\mathcal{B} = (8, 7, 5, 3)$

Dynamic range:  $M = 8 \times 7 \times 5 \times 3 = 840$ , i.e.,  $0 \leq A < M$

$A_{std}$	$A_{RNS}$	$A_{std}$	$A_{RNS}$	$A_{std}$	$A_{RNS}$
0	[0, 0, 0, 0]	9	[1, 2, 4, 0]	18	[2, 4, 3, 0]
1	[1, 1, 1, 1]	10	[2, 3, 0, 1]	19	[3, 5, 4, 1]
2	[2, 2, 2, 2]	11	[3, 4, 1, 2]	20	[4, 6, 0, 2]
3	[3, 3, 3, 0]	12	[4, 5, 2, 0]	21	[5, 0, 1, 0]
4	[4, 4, 4, 1]	13	[5, 6, 3, 1]	22	[6, 1, 2, 1]
5	[5, 5, 0, 2]	14	[6, 0, 4, 2]	23	[7, 2, 3, 2]
6	[6, 6, 1, 0]	15	[7, 1, 0, 0]	24	[0, 3, 4, 0]
7	[7, 0, 2, 1]	16	[0, 2, 1, 1]	25	[1, 4, 0, 1]
8	[0, 1, 3, 2]	17	[1, 3, 2, 2]	26	[2, 5, 1, 2]

## Residue Number System: Example (2/2)

Operands:  $A = 6 = [6, 6, 1, 0]$  and  $B = 16 = [0, 2, 1, 1]$

Addition:

- $(6 + 0) \bmod 8 = 6$
- $(6 + 2) \bmod 7 = 1$
- $(1 + 1) \bmod 5 = 2$
- $(0 + 1) \bmod 3 = 1$

Verification:  $22 = [6, 1, 2, 1]$

Multiplication:

- $(6 \times 0) \bmod 8 = 0$
- $(6 \times 2) \bmod 7 = 5$
- $(1 \times 1) \bmod 5 = 1$
- $(0 \times 1) \bmod 3 = 0$

Verification:  $96 = [0, 5, 1, 0]$

## Residue Number System: Conversions

From standard to RNS:

$$\forall i \quad a_i = A \bmod m_i$$

From RNS to standard:

Using a constructing proof of the Chinese Remainder Theorem (CRT)

$$A = \sum_{i=1}^k a_i M_i |M_i^{-1}|_{m_i} \bmod M$$

where

- $M = \prod_{i=1}^k m_i$ ,  $A < M$
- $M_i = M/m_i$
- $|M_i^{-1}|_{m_i}$  is the inverse of  $M_i$  modulo  $m_i$

## Residue Number System: Summary

Advantages:

- fast addition/subtraction and multiplication (parallel)
- no carry propagation (on the whole format)
- natural way to split large numbers (simple scheduling)
- no order in the elements

Disadvantages:

- difficult comparison (< and >)
- difficult division
- difficult sign test
- difficult magnitude computation