Modern Computer Arithmetic

Paul Zimmermann

INRIA Nancy - Grand Est, France

Microsoft Research, Redmond, June 12, 2009

(common work with Richard Brent, ANU, Canberra, Australia)

Modern Computer Arithmetic

Richard P. Brent and Paul Zimmermann

Version 0.3

History of *Modern Computer Arithmetic*

Started in July 2003, still in progress.

Freely available from
`http://www.loria.fr/~zimmerma/mca/pub226.html`

Current version (0.3) has 221 pages.

Paper version hopefully out in 2010.

The on-line version should remain available.

Yet another book?

Some parts of Knuth vol. 2 are out-of-date: divide and conquer division, Schönhage-Strassen FFT, floating-point algorithms

Most books only cover schoolbook-range algorithms or FFT-range algorithms, not middle-range algorithms

Most books only cover fast multiplication, a few cover fast division, very few cover other algorithms (square root, radix conversion, gcd, . . .)

Most books let the reader deal with carries, rounding errors, . . .

We claim our algorithms can be implemented "out-of-the-box".

Other Related Books (1/2)

The Design and Analysis of Computer Algorithms, Aho, Hopcroft, Ullman, 1974, fast polynomial gcd

The Art of Computer Programming, vol .2, Seminumerical Algorithms, D. E. Knuth, 3rd edition, 1998, many algorithms

Fast Algorithms, A Multitape Turing Machine Implementation, Schönhage, Grotefeld, Vetter, 1994, some nice algorithms

Handbook of Applied Cryptography, Chapter 14, Menezes, van Oorschot, Vanstone, 1997: only covers schoolbook range

Other Related Books (2/2)

Modern Computer Algebra, von zur Gathen and Gerhard, 1999: same spirit for polynomial and power series

Prime Numbers: A Computational Perspective, Chapter 9 (Fast Algorithms for Large-Integer Arithmetic), 2001

Handbook of Elliptic and Hyperelliptic Curve Cryptography, Cohen, Frey, Avanzi, Doche, Lange, Vercauteren, 2005 (chapters 9, 10, 11, 12), mainly covers schoolbook range

Handbook of Floating-Point Arithmetic, Brisebarre, de Dinechin, Jeannerod, Lefèvre, Melquiond, Muller, Revol, Stehlé, Torres, Birkhäuser, 2009-2010, only covers floating-point algorithms

GNU MP Reference Manual, Chapter Algorithms, unknown but very relevant

Why?

Richard Brent has a long experience in designing fast arithmetic algorithms

rpb017, rpb028,rpb032,rpb034,rpb042
rpb043,rpb052,rpb049,rpb052,rpb069

and implementing them (the Fortran MP package).

Paul Zimmermann has a long (but shorter) experience, with contributions to the GNU MP and the GNU MPFR libraries.

Hopefully useful to researchers and engineers in computer arithmetic, developers of arithmetic packages, and also students.

Advertisement Slide

GNU  is an arbitrary precision floating-point library with correct rounding

Written in the C language, download from `www.mpfr.org`

Extends IEEE 754-2008 to *arbitrary precision* and *mathematical functions*

Used by GCC and GFORTRAN for *constant folding*:

```
double x = sin(3.14)
```

Might enable to extend Microsoft Excel to arbitrary precision, while having reproducibility of the results across different platforms

How?

The book contains 4 chapters:

1. Integer Arithmetic
2. The FFT and Modular Arithmetic
3. Floating-Point Arithmetic
4. Newton's Method and Function Evaluation

In each chapter we try to cover the main algorithms.

(Too complex or too technical algorithms in the exercises.)

While writing this book, several questions came to us.

Some questions led to new algorithms, some are still unanswered.

How do you discover a new algorithm?

	operation 1	operation 2
case A	algo X	algo Y
case B	algo Z	???

Plan of the talk

- The Binary Gcd (a new algorithm due to the book)
- Divide and Conquer Division (middle-range algorithm)
- Hensel Division and Montgomery Reduction (better understanding)
- Odd-Even Karatsuba Scheme and Unbalanced Multiplication (new algorithm)
- Svoboda Division (lost algorithm?)
- Half-Montgomery-Reduction (almost new algorithm)
- Batch Multiplication (bonus track)

Disclaimer: this is my view of the book.

The binary gcd (with Stehlé)

We can compute a asymptotically fast gcd of two n-bit integers in $O(M(n)\log n)$ using Knuth-Schönhage's algorithm (proposed by Knuth in 1970 but with a $O(n\log^5 n \log\log n)$ complexity, improved by Schönhage in 1971 to $O(n\log^2 n \log\log n)$)

Because of carries, this fast-gcd algorithm requires a "fix-up procedure", which is tricky to design and implement correctly (most textbooks descriptions are incomplete or wrong)

On the other hand, we know how to divide integers by the least significant bits (LSB).

Question (asked to Damien Stehlé in 2003): can we design an asymptotically fast LSB gcd algorithm?

	division	gcd
classical	$O(M(n))$	$O(M(n)\log n)$
binary	$O(M(n))$	???

Classical fast GCD

Consider $a = 935$ and $b = 714$.

$$a_{i-1} = q_i a_i + a_{i+1}$$

$a_0 = 935$, $a_1 = 714$, $a_2 = 221$, $a_3 = 51$, $a_4 = 17$, $a_5 = 0$.

$$\begin{pmatrix} a_i \\ a_{i-1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & q_i \end{pmatrix} \begin{pmatrix} a_{i+1} \\ a_i \end{pmatrix}$$

$$\begin{pmatrix} a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 4 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} a_5 \\ a_4 \end{pmatrix}$$

Hint: determine the q_i from the MSB of the a_i, and multiply the $\begin{pmatrix} 0 & 1 \\ 1 & q_i \end{pmatrix}$ matrices using $O(M(n))$ multiplication (product tree)

Problem: correctly determine the q_i despite carries.

Classical fast GCD: binary view

935	1110100111
714	1011001010
221	0011011101
51	0000110011
17	0000010001
0	0000000000

Binary division

Definition (Binary Division)
Let $a, b \in \mathbb{Z}$ with $\nu_2(b) > \nu_2(a)$.
Let $j = \nu_2(b) - \nu_2(a)$.
There is a unique $|q| < 2^j$ such that with $\nu_2(b) < \nu_2(r)$:

$$r = a + q2^{-j}b$$

q is the binary quotient of a by b
r is the binary remainder of a by b

Let $a' = a/2^{\nu_2(a)}$, $b' = b/2^{\nu_2(b)}$.
$\nu_2(a') = \nu_2(b') = 0$
We want $\nu_2(a' + qb') > j$, thus $a' + qb' \equiv 0 \bmod 2^{j+1}$.

$$q \equiv -a'/b' \bmod 2^{j+1}$$

Binary division: example

Let $a = 935 = (1110100111)_2$ with $\nu_2(a) = 0$.
Let $b = 714 = (1011001010)_2$ with $\nu_2(b) = 1 > \nu_2(a)$.

$$q \equiv -935/(714/2) \equiv 1 \bmod 2^2$$

$r = a + 1 \cdot b/2 = 1292 = (10100001100)_2$.

Important note: keep the LSB zeroes!

Binary GCD

Simply iterate binary division until 0.

i	q_i	a_i	a_i(binary)
0		935	1110100111
1	0	714	1011001010
2	1	1292	10100001100
3	1	1360	10101010000
4	1	1632	11001100000
5	1	2176	100010000000
6	-3	0	000000000000

The gcd of a, b is the odd part of the last non-zero term.

Binary GCD: pros and cons

$\oplus$ carries go to the MSB: no "fix-up" needed any more

$\oplus$ $q = a/b \bmod 2^j$ simpler to compute than $q = \lfloor a/b \rfloor$

$\oplus$ same complexity $O(M(n) \log n)$ for the divide and conquer variant as for the classical GCD

$\ominus$ small linear growth of the MSB part (about 10%), precisely analyzed by Brigitte Vallée using *dynamic analysis*

Modular Inverse via Binary GCD

Assume we want to invert b modulo a.

Given a, b, the binary GCD computes:

$$\alpha a + \beta b = 2^k g$$

Assume $\gcd(a, b) = g = 1$:

$$\frac{1}{b} = \frac{\beta}{2^k} \bmod a$$

Note: $2^{-k} \bmod a$ can be computed in $O(M(n))$, much faster than the GCD itself.

Divide and Conquer Division

Precisely described with Karastuba multiplication by Burnikel and Ziegler (1998), previously proposed by Moenck and Borodin (1972) and Jebelean (1997).

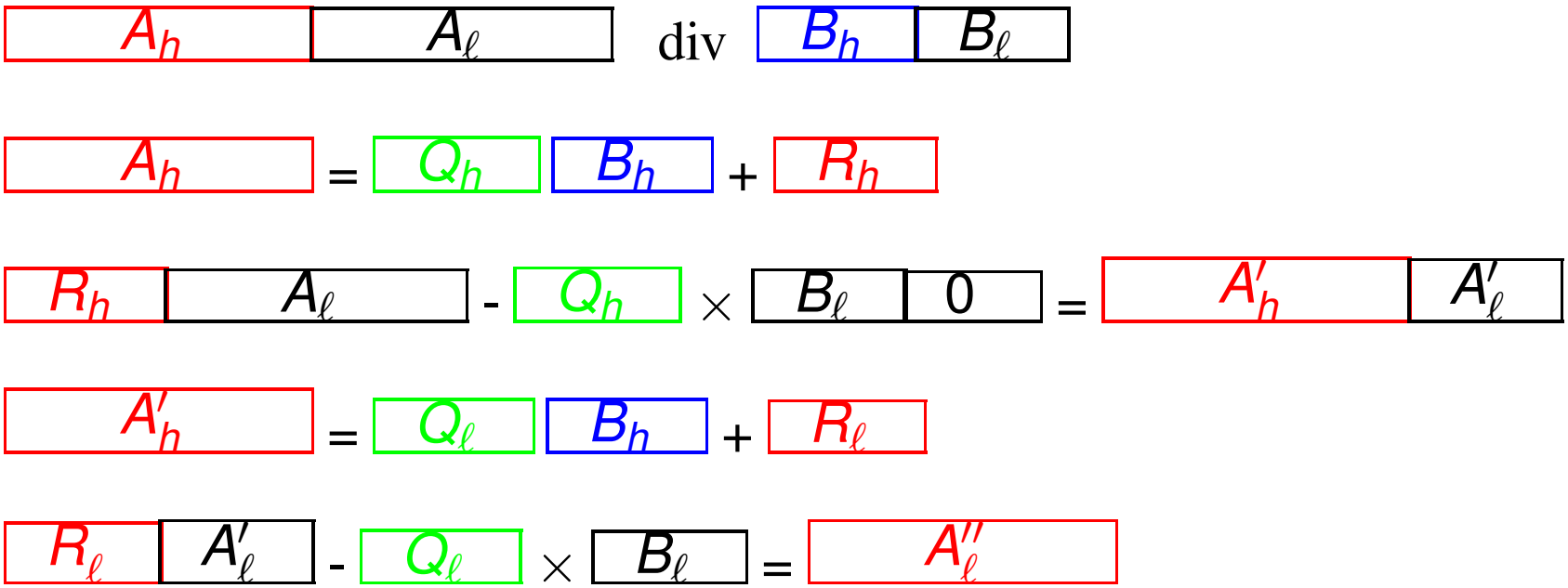
Input: $A = \sum_0^{n+m-1} a_i \beta^i$, $B = \sum_0^{n-1} b_j \beta^j$, B normalized, $n \geq m$

Output: quotient Q and remainder R of A divided by B.

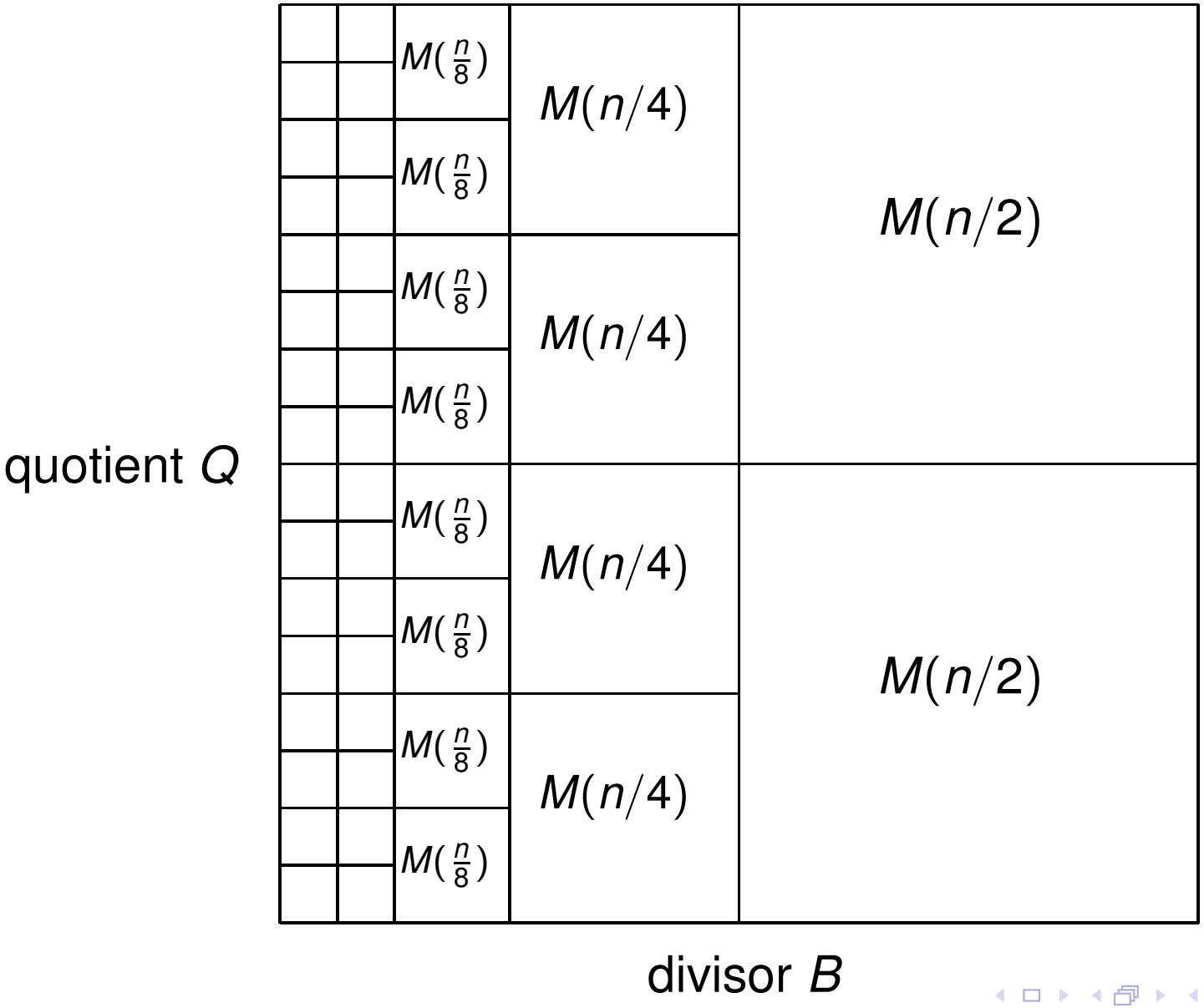
1: if $m < 2$ then return **BasecaseDivRem**(A, B)
2: $k \leftarrow \lfloor \frac{m}{2} \rfloor$, $B_1 \leftarrow B \operatorname{div} \beta^k$, $B_0 \leftarrow B \bmod \beta^k$
3: $(Q_1, R_1) \leftarrow$ **RecursiveDivRem**$(A \operatorname{div} \beta^{2k}, B_1)$
4: $A' \leftarrow R_1 \beta^{2k} + (A \bmod \beta^{2k}) - Q_1 B_0 \beta^k$
5: while $A' < 0$ do $Q_1 \leftarrow Q_1 - 1$, $A' \leftarrow A' + \beta^k B$
6: $(Q_0, R_0) \leftarrow$ **RecursiveDivRem**$(A' \operatorname{div} \beta^k, B_1)$
7: $A'' \leftarrow R_0 \beta^k + (A' \bmod \beta^k) - Q_0 B_0$
8: while $A'' < 0$ do $Q_0 \leftarrow Q_0 - 1$, $A'' \leftarrow A'' + B$
9: **return** $Q := Q_1 \beta^k + Q_0$, $R := A''$.

Ready to implement, with carries

Divide and Conquer Division: Graphical View



Divide and Conquer Division: Graphical View



Divide and Conquer Division: Complexity

$$D(n) = 2D(n/2) + 2M(n/2)$$

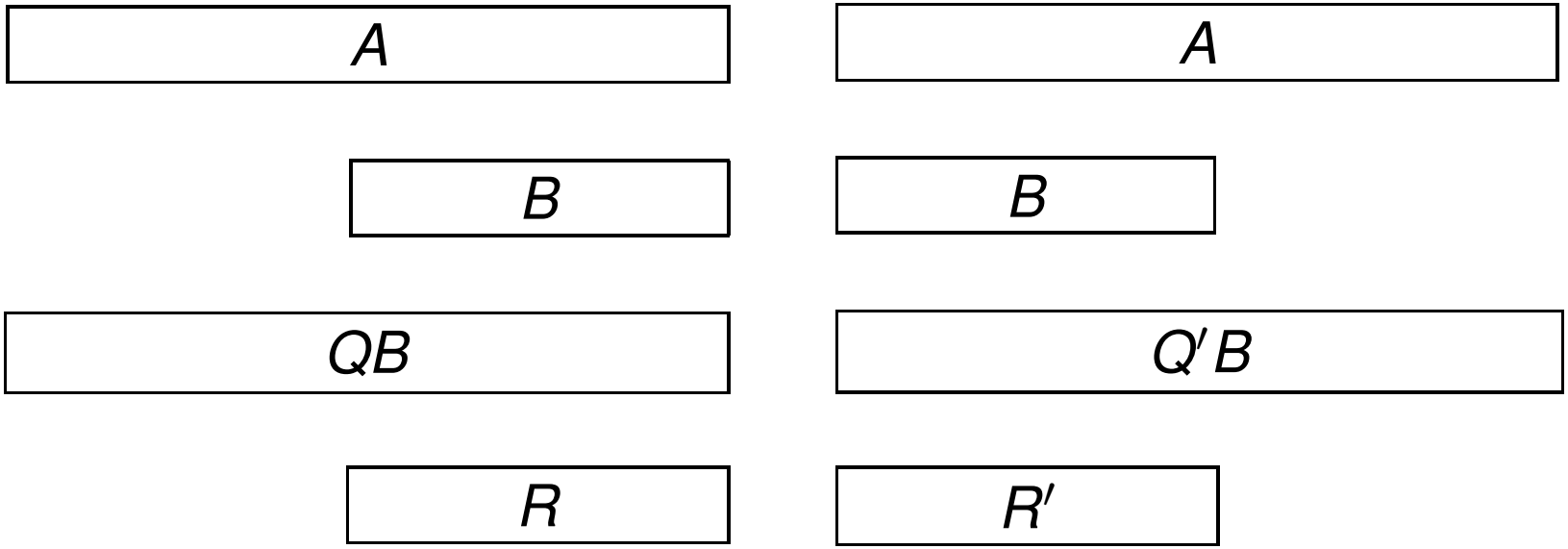
Karatsuba range: $D(n) = 2M(n)$ (best possible $D(n) = M(n)$ with van der Hoeven's algorithm)

Toom-Cook range: $M(n) \approx n^{1.47}$: $D(n) \approx 2.63M(n)$

FFT range: $D(n) = \Theta(M(n) \log n)$

Still the only subquadratic division in GMP 4.3! (Threshold is 27 limbs on a Pentium M.)

Classical vs Hensel Division



classical division (MSB left)

Hensel division (LSB right)

$$A = QB + R$$

$$A = Q'B + R'2^n$$

Hensel Division vs Montgomery Reduction

Hensel Division vs Montgomery Reduction

Hensel division:

$$A = QB + R2^n$$

Quotient is the 2-adic quotient:

$$Q = A/B \bmod 2^n$$

Remainder is the result of Montgomery REDC:

$$R = A/2^n \bmod B$$

Odd-Even Karatsuba Scheme (with Hanrot)

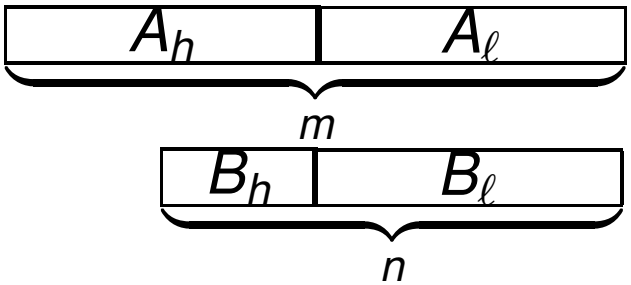
Classical Karatsuba scheme.

$$A = a_{2n-1}\beta^{2n-1} + \cdots + a_1\beta + a_0$$

Write $A = A_h\beta^n + A_\ell$, $B = B_h\beta^n + B_\ell$
Evaluate A_hB_h, $A_\ell B_\ell$, $(A_h + A_\ell)(B_h + B_\ell)$.

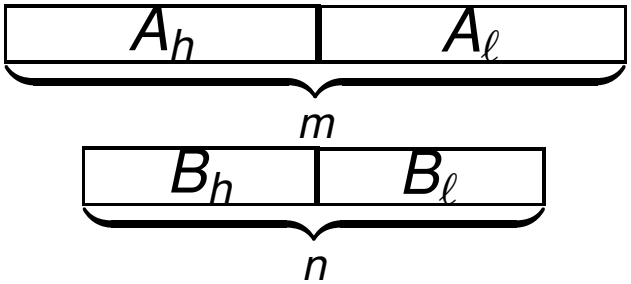
Odd-Even Karatsuba scheme.
Write $A = A_{\text{odd}}\beta + A_{\text{even}}$, $B = B_{\text{odd}}\beta + B_{\text{even}}$
Evaluate $A_{\text{odd}}B_{\text{odd}}$, $A_{\text{even}}B_{\text{even}}$, $(A_{\text{odd}} + A_{\text{even}})(B_{\text{odd}} + B_{\text{even}})$.

Karatsuba for unbalanced operands



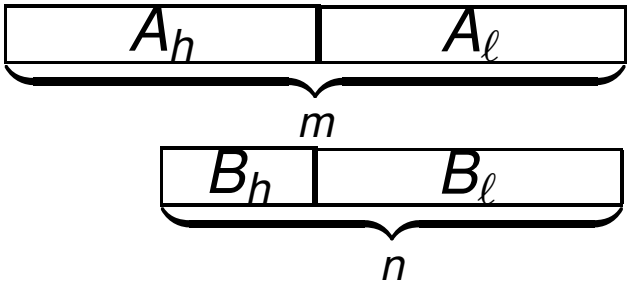
$$K(m,n) = K(m/2, n - m/2) + 2K(m/2)$$

Better to "center" B?



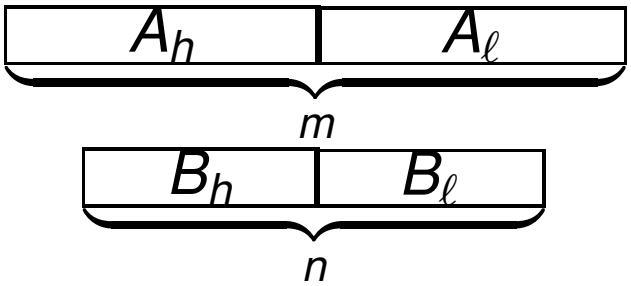
$$K(m,n) = 2K(m/2, n/2) + K(m/2)$$

LSB aligned vs centered



$$K(5,3) = K(2,0) + K(3,3) + K(3,3) = 2 \cdot 7 = 14$$

$$K(3,3) = 2K(2,2) + K(1,1) = 7$$



$$K(5,3) = K(2,2) + K(3,1) + K(3,3) = 3 + 3 + 7 = 13$$

Not always best: $K(6,4) = 17$ LSB aligned, 19 centered

Unbalanced operands with the odd-even scheme

$a_4 a_3 a_2 a_1 a_0 \times b_2 b_1 b_0$

$a_4 a_2 a_0 \times b_2 b_0$

$a_3 a_1 \times b_1$

$a_4 a_3 + a_2 a_1 + a_0 \times b_2 b_1 + b_0$

$$K(m,n) = 2K(\lceil m/2 \rceil, \lceil n/2 \rceil) + K(\lfloor m/2 \rfloor, \lfloor n/2 \rfloor)$$

$$K(3,2) = 2K(2,1) + K(1,1) = 5$$

$$K(5,3) = 2K(3,2) + K(2,1) = 2 \times 5 + 2 = 12$$

We can do even better!

When both m and n are odd, zero-pad b to the LSB:

$a_4|a_3|a_2|a_1|a_0 \times b_2|b_1|b_0|0$

$a_4|a_2|a_0 \times b_1|0$

$a_3|a_1 \times b_2|b_0$

$a_4|a_3 + a_2|a_1 + a_0 \times b_2 + b_1|b_0$

$$K(m,n) = K(\lceil m/2 \rceil, \lfloor n/2 \rfloor) + K(\lfloor m/2 \rfloor, \lceil n/2 \rceil) + K(\lceil m/2 \rceil, \lceil n/2 \rceil)$$

$$K(3,2) = K(2,1) + K(1,1) + K(2,1) = 2 + 1 + 2 = 5$$

$$K(5,3) = K(3,1) + K(2,2) + K(3,2) = 3 + 3 + 5 = 11$$

Montgomery's Multiplication

Idea: $AB \bmod D \quad \Longrightarrow \quad \mathbf{REDC}(A, B) := AB\beta^{-n} \bmod D$

$$A \quad \Longrightarrow \quad \widetilde{A} := A\beta^n$$

Montgomery's Multiplication

Idea: $AB \bmod D \quad \Longrightarrow \quad \mathbf{REDC}(A, B) := AB\beta^{-n} \bmod D$

$$A \quad \Longrightarrow \quad \widetilde{A} := A\beta^n$$

$$\boxed{\widetilde{A} = A\beta^n} \qquad \boxed{\widetilde{B} = B\beta^n}$$

Montgomery's Multiplication

Idea: $AB \bmod D \quad \Longrightarrow \quad \mathbf{REDC}(A, B) := AB\beta^{-n} \bmod D$

$$A \quad \Longrightarrow \quad \widetilde{A} := A\beta^n$$

$$\boxed{\widetilde{A} = A\beta^n} \qquad \boxed{\widetilde{B} = B\beta^n}$$

$$\Downarrow \qquad M(n)$$

$$\boxed{\widetilde{A} \times \widetilde{B} = AB\beta^{2n}}$$

Montgomery's Multiplication

Idea: $AB \bmod D \quad \Longrightarrow \quad \mathbf{REDC}(A,B) := AB\beta^{-n} \bmod D$

$$A \quad \Longrightarrow \quad \widetilde{A} := A\beta^n$$

$$\boxed{\widetilde{A} = A\beta^n} \qquad \boxed{\widetilde{B} = B\beta^n}$$

$$\Downarrow \qquad M(n)$$

$$\boxed{\widetilde{A} \times \widetilde{B} = AB\beta^{2n}}$$

$$\Downarrow \qquad \widetilde{D}(n)$$

$$\boxed{\widetilde{A}\widetilde{B}\beta^{-n} = AB\beta^n = \widetilde{AB} \bmod D}$$

Montgomery's Reduction (schoolbook range)

Input: $C < D^2$, $\mu = -D^{-1} \bmod \beta$ (precomputed)
Output: $R = C\beta^{-n} \bmod D$
for i from 0 to $n-1$ do
  $q_i \leftarrow \mu c_i \bmod \beta$ (quotient selection)
  $C \leftarrow C + q_i D \beta^i$
$R \leftarrow C\beta^{-n}$
if $R \geq \beta^n$ then return $R - D$ else return R (fixup)

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$C = 766\,970\,544\,842\,443\,844 \;\big|\; D = 862\,664\,913$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$C = 766\,970\,544\,842\,443\,844 \quad \Big| \quad D = 862\,664\,913$$

$C = 766\,970\,544\,842\,443\,844$	$D = 862\,664\,913$
	412

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\ \hline & 412 \\ + \underline{355\,417\,944\,156} & \end{array}$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\ \hline & 412 \\ + \underline{355\,417\,944\,156} & \\ 766\,970\,900\,260\,388 & \end{array}$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\ \hline & 412 \\ + \underline{355\,417\,944\,156} & \\ 766\,970\,900\,260\,388 & 924 \end{array}$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|l}
C = 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & 412 \\
+\ \underline{355\,417\,944\,156} & \\
766\,970\,900\,260\,388 & 924 \\
+\ \underline{797\,102\,379\,612} &
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{797\,102\,379\,612} & \\
 & 767\,768\,002\,640 &
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{797\,102\,379\,612} & \\
 & 767\,768\,002\,640 & 720
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|r}
C = 766\,970\,544\,842\,443\,\color{red}{844} & D = 862\,664\,\color{red}{913} \\
\hline
 & 412 \\
+\,\underline{355\,417\,944\,156} & \\
766\,970\,900\,260\,\color{red}{388} & 924 \\
+\,\underline{797\,102\,379\,612} & \\
767\,768\,002\,\color{red}{640} & 720 \\
+\,\underline{621\,118\,737\,360} &
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{\quad 355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{\quad 797\,102\,379\,612} & \\
 & 767\,768\,002\,640 & 720 \\
+ & \underline{621\,118\,737\,360} & \\
 & 1\,388\,886\,740 &
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{\quad 355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{\quad 797\,102\,379\,612} & \\
 & 767\,768\,002\,640 & \quad 720 \\
+ & \underline{621\,118\,737\,360} & \\
 & 1\,388\,886\,740 & \\
- & \underline{862\,664\,913} & -1
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{\quad 355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{\quad 797\,102\,379\,612} & \\
 & 767\,768\,002\,640 & \quad 720 \\
+ & \underline{621\,118\,737\,360} & \\
 & 1\,388\,886\,740 & \\
- & \underline{862\,664\,913} & -1 \\
 & 526\,221\,827 &
\end{array}
$$

Montgomery's Multiplication: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{rl|l}
C = & 766\,970\,544\,842\,443\,844 & D = 862\,664\,913 \\
\hline
 & & \quad 412 \\
 & + \underline{\quad 355\,417\,944\,156} & \\
 & 766\,970\,900\,260\,388 & \quad 924 \\
 & + \underline{\quad 797\,102\,379\,612} & \\
 & 767\,768\,002\,640 & \quad 720 \\
+ & \underline{621\,118\,737\,360} & \\
 & 1\,388\,886\,740 & \\
- & \underline{862\,664\,913} & -1 \\
 & 526\,221\,827 &
\end{array}
$$

$$C + 720924412 \times D = 10^9 \cdot 1388886740 = 10^9(D + 526221827)$$

Classical vs Montgomery Quadratic Division

- quotient selection is expensive (128 bits $\div$ 64 bits)

  64 bits $\times$ 64 bits mod 2^{64}: cheap

Classical vs Montgomery Quadratic Division

- quotient selection is expensive (128 bits $\div$ 64 bits)

  64 bits $\times$ 64 bits mod 2^{64}: cheap

- up to $2n$ repair steps per division

  at most one final repair step

Classical vs Montgomery Quadratic Division

- quotient selection is expensive (128 bits $\div$ 64 bits)

  64 bits $\times$ 64 bits mod 2^{64}: cheap

- up to $2n$ repair steps per division

  at most one final repair step

- difficult branch prediction

  no branch inside loop

Classical vs Montgomery Quadratic Division

- quotient selection is expensive (128 bits $\div$ 64 bits)
  64 bits $\times$ 64 bits mod 2^{64}: cheap
- up to $2n$ repair steps per division
  at most one final repair step
- difficult branch prediction
  no branch inside loop
- dependency between repair step and next loop: c_{n+j}, c_{n+j-1}
  still holds for c_i

Classical vs Montgomery Quadratic Division

- quotient selection is expensive (128 bits $\div$ 64 bits)
  64 bits $\times$ 64 bits mod 2^{64}: cheap
- up to $2n$ repair steps per division
  at most one final repair step
- difficult branch prediction
  no branch inside loop
- dependency between repair step and next loop: c_{n+j}, c_{n+j-1}
  still holds for c_i
- and it is quadratic ...
  still holds

Svoboda Division

When dividing by $D = \overbrace{1,000}^{d_{n-1}}, \ldots$, quotient selection is easy:

$$\lfloor \frac{c_{n+j}\beta + c_{n+j-1}}{d_{n-1}} \rfloor = c_{n+j}$$

Svoboda's idea: force $d_{n-1} = \beta$!

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$766\,970\,544\,842\,443\,844 \;\big|\; 1000\,691\,299\,080$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
	766

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$\begin{array}{rl|l} & 766\,970\,544\,842\,443\,844 & 1000\,691\,299\,080 \\ \cline{3-3} - & \underline{766\,529\,535\,095\,280} & 766 \end{array}$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$\begin{array}{rl|l} & 766\,970\,544\,842\,443\,844 & 1000\,691\,299\,080 \\ \cline{3-3} - & \underline{766\,529\,535\,095\,280} & 766 \\ & 441\,009\,747\,163\,844 & \end{array}$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$\begin{array}{rl|l} & 766\,970\,544\,842\,443\,844 & 1000\,691\,299\,080 \\ \hline -\ & \underline{766\,529\,535\,095\,280} & 766 \\ & 441\,009\,747\,163\,844 & \quad 441 \end{array}$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$
\begin{array}{rl|l}
 & 766\,970\,544\,842\,443\,844 & 1000\,691\,299\,080 \\
\cline{3-3}
- & \underline{766\,529\,535\,095\,280} & 766 \\
 & 441\,009\,747\,163\,844 & \quad 441 \\
- & \underline{441\,304\,862\,894\,280} & \\
 & & \\
\end{array}
$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$$
\begin{array}{rl|l}
 & 766\,970\,544\,842\,443\,844 & 1000\,691\,299\,080 \\
\cline{3-3}
- & \underline{766\,529\,535\,095\,280} & 766 \\
 & 441\,009\,747\,163\,844 & \quad 441 \\
- & \underline{441\,304\,862\,894\,280} & \\
 & -\,295\,115\,730\,436 &
\end{array}
$$

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ \underline{766\,529\,535\,095\,280}$	766
$441\,009\,747\,163\,844$	441
$-\ \underline{441\,304\,862\,894\,280}$	
$-\ 295\,115\,730\,436$	
$+\ \underline{1000\,691\,299\,080}$	-1

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ \underline{766\,529\,535\,095\,280}$	766
$441\,009\,747\,163\,844$	441
$-\ \underline{441\,304\,862\,894\,280}$	
$-\ 295\,115\,730\,436$	
$+\ \underline{1000\,691\,299\,080}$	-1
$705\,575\,568\,644$	

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ \underline{766\,529\,535\,095\,280}$	766
$441\,009\,747\,163\,844$	441
$-\ \underline{441\,304\,862\,894\,280}$	
$-\ 295\,115\,730\,436$	
$+\ \underline{1000\,691\,299\,080}$	-1
$705\,575\,568\,644$	818

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ 766\,529\,535\,095\,280$	766
$441\,009\,747\,163\,844$	441
$-\ 441\,304\,862\,894\,280$	
$-\ 295\,115\,730\,436$	
$+\ 1000\,691\,299\,080$	-1
$705\,575\,568\,644$	818
$-\ 705\,659\,898\,834$	

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ \underline{766\,529\,535\,095\,280}$	766
$441\,009\,747\,163\,844$	441
$-\ \underline{441\,304\,862\,894\,280}$	
$-\ 295\,115\,730\,436$	
$+\ \underline{1000\,691\,299\,080}$	-1
$705\,575\,568\,644$	818
$-\ \underline{705\,659\,898\,834}$	
$-\ 084\,330\,190$	

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ 766\,529\,535\,095\,280$	766
$441\,009\,747\,163\,844$	441
$-\ 441\,304\,862\,894\,280$	
$-\ 295\,115\,730\,436$	
$+\ 1000\,691\,299\,080$	-1
$705\,575\,568\,644$	818
$-\ 705\,659\,898\,834$	
$-\ 084\,330\,190$	
$+\ 862\,664\,913$	-1

Svoboda Division: An Example

Precompute $D' = 1160 \cdot D = 1000\,691\,299\,080$

$766\,970\,544\,842\,443\,844$	$1000\,691\,299\,080$
$-\ 766\,529\,535\,095\,280$	766
$441\,009\,747\,163\,844$	441
$-\ 441\,304\,862\,894\,280$	
$-\ 295\,115\,730\,436$	
$+\ 1000\,691\,299\,080$	-1
$705\,575\,568\,644$	818
$-\ 705\,659\,898\,834$	
$-\ 084\,330\,190$	
$+\ 862\,664\,913$	-1
$778\,334\,723$	

$$C = (766440 \times 1160 + 817)D + 778334723$$

Svoboda Division:

$$C = Q(kD) + qD + R$$

- quotient selection becomes trivial (except last step)
- smaller repair probability, since d_{n-1} larger
- very interesting when quotient is not needed

How to Use Svoboda Division?

- either choose D such that $d_{n-1} = \beta$
- or work modulo kD of $n + 1$ words
- or perform a last ordinary division step

Montgomery-Svoboda Division

Montgomery Reduction:
Input: $C < D^2$, $\mu = -D^{-1} \bmod \beta$ (precomputed)
Output: $R = C\beta^{-n} \bmod D$
for i from 0 to $n-1$ do
  $q_i \leftarrow \mu c_i \bmod \beta$ (quotient selection)
  $C \leftarrow C + q_i D \beta^i$
$R \leftarrow C\beta^{-n}$
if $R \geq \beta^n$ then return $R - D$ else return R

Montgomery-Svoboda Division

Montgomery Reduction:
Input: $C < D^2$, $\mu = -D^{-1} \bmod \beta$ (precomputed)
Output: $R = C\beta^{-n} \bmod D$
for i from 0 to $n-1$ do
  $q_i \leftarrow \mu c_i \bmod \beta$ (quotient selection)
  $C \leftarrow C + q_i D \beta^i$
$R \leftarrow C\beta^{-n}$
if $R \geq \beta^n$ then return $R - D$ else return R

Montgomery-Svoboda Reduction:
Input: $C < D^2$, $\mu = -D^{-1} \bmod \beta$ (precomputed), μD
Output: $R = C\beta^{-n} \bmod D$
for i from 0 to $n-2$ do
  $q_i \leftarrow c_i \bmod \beta$ (trivial quotient selection)
  $C \leftarrow C + q_i(\mu D)\beta^i$
$q_{n-1} \leftarrow \mu c_{n-1} \bmod \beta$
$C \leftarrow C + q_{n-1} D \beta^{n-1}$
$R \leftarrow C\beta^{-n}$
if $R \geq \beta^n$ then return $R - D$ else return R

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$C = 766\,970\,544\,842\,443\,844 \;\Big|\; D' = 19\,841\,292\,999$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$C = 766\,970\,544\,842\,443\,844$	$D' = 19\,841\,292\,999$
	844

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\ \underline{+\ 16\,746\,051\,291\,156} & \hline 844 \end{array}$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\ \underline{+\ 16\,746\,051\,291\,156} & \hline \\ 766\,987\,290\,893\,735 & 844 \end{array}$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$\begin{array}{r|l} C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\ \underline{+\ 16\,746\,051\,291\,156} & \hline \\ 766\,987\,290\,893\,735 & \end{array}$$

$$\begin{array}{r} 844 \\ 735 \end{array}$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|l}
C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\
\quad + \underline{16\,746\,051\,291\,156} & \hline
766\,987\,290\,893\,735 & \qquad\qquad 844 \\
\quad + \underline{14\,583\,350\,354\,265} & \qquad\quad 735 \\
\hline
\end{array}
$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|l}
C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\
\quad + \underline{16\,746\,051\,291\,156} & \hline
766\,987\,290\,893\,735 & \qquad\qquad 844 \\
\quad + \underline{14\,583\,350\,354\,265} & \qquad\qquad 735 \\
\hline
781\,570\,641\,248 &
\end{array}
$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$C = 766\,970\,544\,842\,443\,844$	$D' = 19\,841\,292\,999$
$+\ 16\,746\,051\,291\,156$	844
$766\,987\,290\,893\,735$	735
$+\ 14\,583\,350\,354\,265$	
$781\,570\,641\,248$	704

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|l}
C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\
\quad + \underline{16\,746\,051\,291\,156} & \hline
766\,987\,290\,893\,735 & 844 \\
\quad + \underline{14\,583\,350\,354\,265} & 735 \\
\hline
781\,570\,641\,248 & 704 \\
\quad + \underline{607\,316\,098\,752} &
\end{array}
$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$$
\begin{array}{r|l}
C = 766\,970\,544\,842\,443\,844 & D' = 19\,841\,292\,999 \\
\quad + \underline{16\,746\,051\,291\,156} & \hline
766\,987\,290\,893\,735 & 844 \\
\quad + \underline{14\,583\,350\,354\,265} & 735 \\
\hline
781\,570\,641\,248 & 704 \\
\quad + \underline{607\,316\,098\,752} & \\
1\,388\,886\,740 &
\end{array}
$$

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$C = 766\,970\,544\,842\,443\,844$	$D' = 19\,841\,292\,999$
$+\ 16\,746\,051\,291\,156$	844
$766\,987\,290\,893\,735$	735
$+\ 14\,583\,350\,354\,265$	
$781\,570\,641\,248$	704
$+\ 607\,316\,098\,752$	
$1\,388\,886\,740$	
$-\ 862\,664\,913$	-1

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$C = 766\,970\,544\,842\,443\,844$	$D' = 19\,841\,292\,999$
$+\ 16\,746\,051\,291\,156$	844
$766\,987\,290\,893\,735$	735
$+\ 14\,583\,350\,354\,265$	
$781\,570\,641\,248$	704
$+\ 607\,316\,098\,752$	
$1\,388\,886\,740$	
$-\ 862\,664\,913$	-1
$526\,221\,827$	

Montgomery-Svoboda: An Example

Precompute $\mu = -1/913 \bmod 1000 = 23$.

$C = 766\,970\,544\,842\,443\,844$	$D' = 19\,841\,292\,999$
$+\ 16\,746\,051\,291\,156$	844
$766\,987\,290\,893\,735$	735
$+\ 14\,583\,350\,354\,265$	
$781\,570\,641\,248$	704
$+\ 607\,316\,098\,752$	
$1\,388\,886\,740$	
$-\ 862\,664\,913$	-1
$526\,221\,827$	

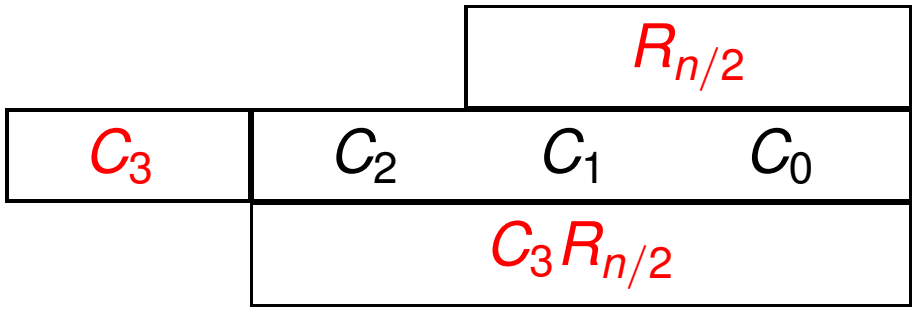
$$C + (23 \times 735844 + 704000000)D = 10^9(D + 526221827)$$

Subquadratic Svoboda Division

Svoboda Division: $kD = \beta^{n+1} + R$, $0 \leq R < D < \beta^n$.
Could also choose $kD = \beta^{n+1} - R$:

$$\beta^{n+1} \equiv R \bmod D$$

Generalization: $\beta^{n+i} \equiv R_i \bmod D$
Take $i = n/2$: reduce from $2n$ to $\frac{3}{2}n$ words in $M(n, n/2)$:



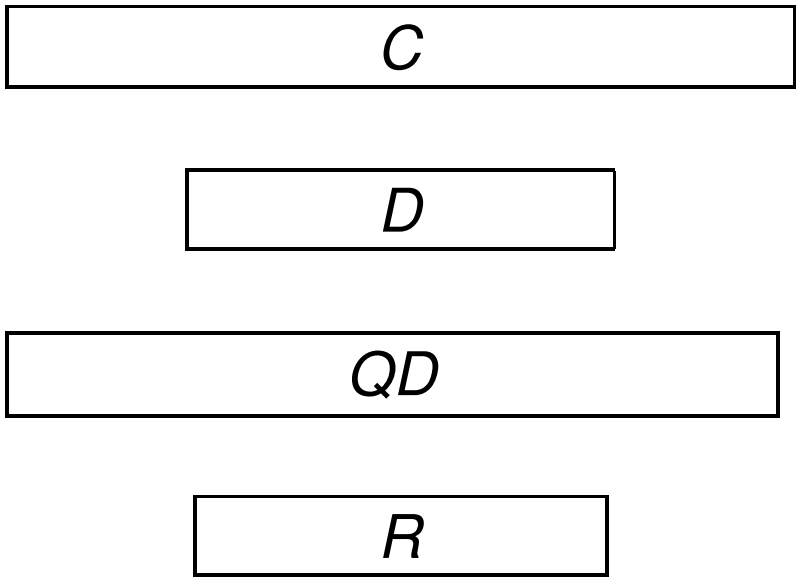
Hasenplaugh, Gaubatz, Gopal, *Fast Modular Reduction*, ARITH'18, 2007.

Half-Montgomery-Reduction

Almost discovered while writing the book (cf last slide of talk at CECC'08).

Bipartite Modular Multiplication Method, Kaihara and Takagi, IEEE TC, 2008.

Idea: half classical divide, half Montgomery reduction.



$$C = QD + R\beta^{n/2}$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705}$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

$$\boxed{000, 444, 145, 552, 584, 651, 444, 705}$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

$$\boxed{000, 444, 145, 552, 584, 651, 444, 705} + 195 \cdot D$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$868, 323, 539, 104, 235, 651, 444, 705$	$-(871 \cdot 10^9) \cdot D$
$000, 444, 145, 552, 584, 651, 444, 705$	$+195 \cdot D$
$000, 444, 145, 746, 886, 008, 210, 000$	

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

$$\boxed{000, 444, 145, 552, 584, 651, 444, 705} + 195 \cdot D$$

$$\boxed{000, 444, 145, 746, 886, 008, 210, 000} - (445 \cdot 10^6) \cdot D$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by $D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

$$\boxed{000, 444, 145, 552, 584, 651, 444, 705} + 195 \cdot D$$

$$\boxed{000, 444, 145, 746, 886, 008, 210, 000} - (445 \cdot 10^6) \cdot D$$

$$\boxed{000, 000, 740, 086, 575, 463, 210, 000}$$

Bipartite Modular Multiplication: an Example

Divide $868, 323, 539, 104, 235, 651, 444, 705$ by
$D = 996, 417, 214, 181$

$$\boxed{868, 323, 539, 104, 235, 651, 444, 705} - (871 \cdot 10^9) \cdot D$$

$$\boxed{000, 444, 145, 552, 584, 651, 444, 705} + 195 \cdot D$$

$$\boxed{000, 444, 145, 746, 886, 008, 210, 000} - (445 \cdot 10^6) \cdot D$$

$$\boxed{000, 000, 740, 086, 575, 463, 210, 000} + (590 \cdot 10^3) \cdot D$$

Bipartite Modular Multiplication: an Example

Divide $868,323,539,104,235,651,444,705$ by
$D = 996,417,214,181$

$868,323,539,104,235,651,444,705$	$-(871 \cdot 10^9) \cdot D$
$000,444,145,552,584,651,444,705$	$+195 \cdot D$
$000,444,145,746,886,008,210,000$	$-(445 \cdot 10^6) \cdot D$
$000,000,740,086,575,463,210,000$	$+(590 \cdot 10^3) \cdot D$
$000,001,327,972,731,830,000,000$	

Bipartite Modular Multiplication: an Example

Divide $868,323,539,104,235,651,444,705$ by
$D = 996,417,214,181$

$868,323,539,104,235,651,444,705$	$-(871 \cdot 10^9) \cdot D$
$000,444,145,552,584,651,444,705$	$+195 \cdot D$
$000,444,145,746,886,008,210,000$	$-(445 \cdot 10^6) \cdot D$
$000,000,740,086,575,463,210,000$	$+(590 \cdot 10^3) \cdot D$
$000,001,327,972,731,830,000,000$	$-D \cdot 10^6$

Bipartite Modular Multiplication: an Example

Divide $868,323,539,104,235,651,444,705$ by
$D = 996,417,214,181$

$868,323,539,104,235,651,444,705$	$-(871 \cdot 10^9) \cdot D$
$000,444,145,552,584,651,444,705$	$+195 \cdot D$
$000,444,145,746,886,008,210,000$	$-(445 \cdot 10^6) \cdot D$
$000,000,740,086,575,463,210,000$	$+(590 \cdot 10^3) \cdot D$
$000,001,327,972,731,830,000,000$	$-D \cdot 10^6$
$000,000,331,555,517,649,000,000$	

Bipartite Modular Multiplication: an Example

Divide $868,323,539,104,235,651,444,705$ by
$D = 996,417,214,181$

$868,323,539,104,235,651,444,705$	$-(871 \cdot 10^9) \cdot D$
$000,444,145,552,584,651,444,705$	$+195 \cdot D$
$000,444,145,746,886,008,210,000$	$-(445 \cdot 10^6) \cdot D$
$000,000,740,086,575,463,210,000$	$+(590 \cdot 10^3) \cdot D$
$000,001,327,972,731,830,000,000$	$-D \cdot 10^6$
$000,000,331,555,517,649,000,000$	

$$\frac{868323539104235651444705}{10^6} \equiv 331,555,517,649 \bmod D$$

Both reduction can be performed independently!

Both reduction can be performed independently!

Step 1a: compute the classical quotient from upper n words $868, 323, 539, 104, 235, 651, 444, 705 \div (10^6 D) = 871, 445$

Both reduction can be performed independently!

Step 1a: compute the classical quotient from upper n words
$868, 323, 539, 104, 235, 651, 444, 705 \div (10^6 D) = 871, 445$

Step 1b: get the Montgomery quotient from lower $n/2$ words
$868, 323, 539, 104, 235, 651, 444, 705 \cdot \mu \equiv 590, 195 \bmod 10^6$
($\mu = -1/D \bmod 10^6$)

Both reduction can be performed independently!

Step 1a: compute the classical quotient from upper n words
$868, 323, 539, 104, 235, 651, 444, 705 \div (10^6 D) = 871, 445$

Step 1b: get the Montgomery quotient from lower $n/2$ words
$868, 323, 539, 104, 235, 651, 444, 705 \cdot \mu \equiv 590, 195 \bmod 10^6$
$(\mu = -1/D \bmod 10^6)$

Step 2: apply the classical quotient
$868, 323, 539, 104, 235, 651, 444, 705 - (871, 445 \cdot 10^6)D =$
$739, 892, 274, 106, 444, 705$

Both reduction can be performed independently!

Step 1a: compute the classical quotient from upper n words
$868, 323, 539, 104, 235, 651, 444, 705 \div (10^6 D) = 871, 445$

Step 1b: get the Montgomery quotient from lower $n/2$ words
$868, 323, 539, 104, 235, 651, 444, 705 \cdot \mu \equiv 590, 195 \bmod 10^6$
$(\mu = -1/D \bmod 10^6)$

Step 2: apply the classical quotient
$868, 323, 539, 104, 235, 651, 444, 705 - (871, 445 \cdot 10^6)D =$
$739, 892, 274, 106, 444, 705$

Step 3: apply the Montgomery quotient
$739, 892, 274, 106, 444, 705 + 590, 195 \cdot D =$
$1, 327, 972, 731, 830, 000, 000$

Step 4: fixup if needed.

Remark: Steps 1a and 2 can be merged.

"Batch Multiplication" (with Kruppa)

Context: in some applications (e.g., stage 2 of ECM), we have points $(x_1 :: z_1), \ldots, (x_k :: z_k)$ on an elliptic curve modulo an integer n, and we want to find a match between them.

$$\prod_{i<j}(x_i z_j - x_j z_i)$$

requires $\approx \frac{3}{2}k^2$ modular products.

It is better to normalize $(x_i :: z_i)$ to $(x_i/z_i :: 1)$, and compute

$$\prod_{i<j}(x_i/z_i - x_j/z_j)$$

with only $\approx \frac{1}{2}k^2$ modular products.

The x_i/z_i can be computed in k modular inversions and k modular multiplications ...

Montgomery's batch inversion

. . . or better in 1 modular inversion and $3k - 3$ modular multiplications using Montgomery's "batch inversion".

Montgomery's batch inversion

. . . or better in 1 modular inversion and $3k - 3$ modular multiplications using Montgomery's "batch inversion".

Input: $z_1, z_2, \ldots, z_k \bmod n$
Output: $1/z_1, \ldots, 1/z_k \bmod n$
(All operations are performed modulo n.)
1. $s_1 = z_1$, $s_2 \leftarrow z_1 z_2, \ldots, s_k \leftarrow z_1 \ldots z_k$ ($k - 1$ mults)
2. $t_k \leftarrow 1/s_k$ (1 inversion)
3. $1/z_k = s_{k-1} t_k, \ldots, 1/z_j = s_{j-1} t_j$, $t_{j-1} = t_j z_j$ ($2k - 2$ mults)

Total 1 inversion, $3k - 3$ multiplications (plus k mults for x_j/z_j)

Batch multiplication

Idea: Instead of normalizing z to 1, normalize to $Z = z_1 z_2 \dots z_k$.

Example ($k = 3$): $(x_1 :: z_1) \to (x_1 z_2 z_3 :: Z)$, $(x_2 :: z_2) \to (z_1 x_2 z_3 :: Z)$, $(x_3 :: z_3) \to (z_1 z_2 x_3 :: Z)$.

1. $\ell_2 = z_1, \ell_3 = z_1 z_2, \dots, \ell_k = z_1 \dots z_{k-1}$ ($k - 2$ mults)
2. $r_{k-1} = z_k, r_{k-2} = z_{k-1} z_k, \dots, r_1 = z_2 \dots z_k$ ($k - 2$ mults)
3. $x_1 r_1, \ell_2 x_2 r_2, \dots, \ell_{k-1} x_{k-1} r_{k-1}, \ell_k r_k$ ($2k - 2$ mults)

Total $4k - 6$ multiplications (saved 1 inversion and 3 mults)