Faster multiprecision integer division

William Hart

June 22, 2015

Multiprecision integer division

Multiprecision integer division

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$

Multiprecision integer division

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$
- Compute:
  - Approximate quotient $Q^\star$

Multiprecision integer division

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$
- Compute:
  - Approximate quotient $Q^\star$
  - Exact quotient Q

Multiprecision integer division

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$
- Compute:
  - Approximate quotient $Q^\star$
  - Exact quotient Q
  - Quotient Q and remainder R

Machine primitive

- Machine does 2×1 division

Machine primitive

- Machine does 2×1 division
- $< u_1, u_0 >$ by $< v_0 >$, quotient and remainder both 1 limb

Machine primitive

- Machine does 2×1 division
- $< u_1, u_0 >$ by $< v_0 >$, quotient and remainder both 1 limb
- Require v_0 *normalised*

Machine primitive

- Machine does 2×1 division
- $< u_1, u_0 >$ by $< v_0 >$, quotient and remainder both 1 limb
- Require v_0 *normalised*
- If $B = 2^{32}$ or 2^{64} then $B < v_0 \leq B/2$

Machine primitive

- Machine does 2×1 division
- $< u_1, u_0 >$ by $< v_0 >$, quotient and remainder both 1 limb
- Require v_0 *normalised*
- If $B = 2^{32}$ or 2^{64} then $B < v_0 \leq B/2$
- Also require $u_1 < v_0$

Schoolbook algorithm

- Shift A and D left so that D is normalised

Schoolbook algorithm

- Shift A and D left so that D is normalised
- Ensure top n limbs of A are less than D

Schoolbook algorithm

- Shift A and D left so that D is normalised
- Ensure top n limbs of A are less than D
- $i \leftarrow m - n + 1$
- while $i \geq 0$
  - $q_i \leftarrow$ quotient of $< a_{n+i}, a_{n+i-1} >$ by $< d_{n-1} >$
  - $A \leftarrow A - q_i D B^i$
  - $i \leftarrow i - 1$

Schoolbook algorithm

- Shift A and D left so that D is normalised
- Ensure top n limbs of A are less than D
- $i \leftarrow m - n + 1$
- while $i \geq 0$
  - $q_i \leftarrow$ quotient of $< a_{n+i}, a_{n+i-1} >$ by $< d_{n-1} >$
  - $A \leftarrow A - q_i D B^i$
  - $i \leftarrow i - 1$
- Shift remainder right

Problems

- May have $< a_{n+i}, a_{n+i-1} > \geq d_{n-1}B$

Problems

- May have $< a_{n+i}, a_{n+i-1} > \geq d_{n-1}B$
- q_iD may be too large

Problems

- May have $< a_{n+i}, a_{n+i-1} > \geq d_{n-1}B$
- q_iD may be too large
- Need tests and adjustments for both cases

Standard improvement No. 1

- Multiply by precomputed inverse ν of d_{n-1} instead of division

Standard improvement No. 1

- Multiply by precomputed inverse ν of d_{n-1} instead of division
  - 1986 Barret used B^{2n}/d

Standard improvement No. 1

- Multiply by precomputed inverse ν of d_{n-1} instead of division
    - 1986 Barret used B^{2n}/d
    - 1986 Beame, Cook, Hoover suggested using an under approximation of the inverse

Standard improvement No. 1

- Multiply by precomputed inverse ν of d_{n-1} instead of division
  - 1986 Barret used B^{2n}/d
  - 1986 Beame, Cook, Hoover suggested using an under approximation of the inverse
  - 1951 Wallace, diode transistor circuits

Standard improvement No. 1

- Multiply by precomputed inverse ν of d_{n-1} instead of division
  - 1986 Barret used B^{2n}/d
  - 1986 Beame, Cook, Hoover suggested using an under approximation of the inverse
  - 1951 Wallace, diode transistor circuits
  - 2000 BC, Babylonian “Clay tablet” PC’s

Precomputed inverses

- 1994, Granlund, Montgomery suggested

$$\nu = \lfloor (B^2 - 1)/d_{n-1} \rfloor - B$$

Precomputed inverses

- 1994, Granlund, Montgomery suggested

$$\nu = \lfloor (B^2 - 1)/d_{n-1} \rfloor - B$$

- One mulhigh, one mul

Precomputed inverses

- 1994, Granlund, Montgomery suggested

$$\nu = \lfloor (B^2 - 1)/d_{n-1} \rfloor - B$$

- One mulhigh, one mul
- At most 3 correction steps for 2×1 division

Precomputed inverses

- 1994, Granlund, Montgomery suggested

$$\nu = \lfloor (B^2 - 1)/d_{n-1} \rfloor - B$$

- One mulhigh, one mul
- At most 3 correction steps for 2×1 division

Additional problem:

- Quotient limb may now be too small or too large!!

Möller-Granlund (2011)

- Same precomputed inverse

Möller-Granlund (2011)

- Same precomputed inverse
- Algorithm improved to one mul, one mullow

Möller-Granlund (2011)

- Same precomputed inverse
- Algorithm improved to one mul, one mullow
- One correction with unpredicted branch

Möller-Granlund (2011)

- Same precomputed inverse
- Algorithm improved to one mul, one mullow
- One correction with unpredicted branch
- One very unlikely correction

Improvement No. 2

Problem : multiprecision corrections *very* expensive

Improvement No. 2

Problem : multiprecision corrections *very* expensive

- Idea : use 3×2 divisions to reduce probability of MP-correction

Improvement No. 2

Problem : multiprecision corrections *very* expensive

- Idea : use 3×2 divisions to reduce probability of MP-correction
- Möller-Granlund give 3×2 version of their algorithm

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1) \rfloor - B$

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1) \rfloor - B$
- Because we used $d_{n-1}+1$, quotient is never too large

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1) \rfloor - B$
- Because we used $d_{n-1}+1$, quotient is never too large
- Approximate quotient can be done with *no* corrections

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1) \rfloor - B$
- Because we used $d_{n-1}+1$, quotient is never too large
- Approximate quotient can be done with *no* corrections
- 3×2 divapprox possible with this precomputed inverse

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1) \rfloor - B$
- Because we used $d_{n-1}+1$, quotient is never too large
- Approximate quotient can be done with *no* corrections
- 3×2 divapprox possible with this precomputed inverse
- Multiprecision corrections rare, always in same direction

Our precomputed inverse

- Use $\nu = \lfloor B^2/(d_{n-1}+1)\rfloor - B$
- Because we used $d_{n-1}+1$, quotient is never too large
- Approximate quotient can be done with *no* corrections
- 3×2 divapprox possible with this precomputed inverse
- Multiprecision corrections rare, always in same direction
- Moreover, MP corrections in *same* direction as 3×2 corrections

Timings (MP divappr)

limbs	*GMP 6.0.0a*	*Our code*
3	$6.7e-8s$	$5.7e-8s$
4	$8.8e-8s$	$7.5e-8s$
6	$1.36e-7s$	$1.17e-7s$
7	$1.85e-7s$	$1.41e-7s$
9	$2.27e-7s$	$1.89e-7s$
11	$2.98e-7s$	$2.38e-7s$
15	$4.35e-7s$	$3.64e-7s$
19	$6.07e-7s$	$4.89e-7s$
21	$7.05e-7s$	$5.96e-7s$
27	$1.05e-6s$	$8.7e-7s$
33	$1.45e-6s$	$1.2e-6s$
37	$1.72e-6s$	$1.48e-6s$

Problem

- 3×2 divapprox good for small MP divisions

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder
- Idea : divide by $d_{n-1} + 1$ first, then adjust

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder
- Idea : divide by $d_{n-1} + 1$ first, then adjust
- Yields 3×2 divrem with *same* precomputed inverse

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder
- Idea : divide by $d_{n-1} + 1$ first, then adjust
- Yields 3×2 divrem with *same* precomputed inverse
- One correction with unpredicted branch, one unlikely correction

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder
- Idea : divide by $d_{n-1} + 1$ first, then adjust
- Yields 3×2 divrem with *same* precomputed inverse
- One correction with unpredicted branch, one unlikely correction
- One additional 2 limb subtraction over Möller-Granlund

Problem

- 3×2 divapprox good for small MP divisions
- ... but MP corrections eventually become significant
- Need 3×2 division with *exact* quotient and remainder
- Idea : divide by $d_{n-1} + 1$ first, then adjust
- Yields 3×2 divrem with *same* precomputed inverse
- One correction with unpredicted branch, one unlikely correction
- One additional 2 limb subtraction over Möller-Granlund
- No real time difference to Möller-Granlund for MP division

Outline

Improvement 3: divapprox

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$

  Let $s \leftarrow m - n + 1$
- Compute:
  - Approx. quotient $Q^\star =< q_{s-1}, \ldots, q_1, q_0 >$

Improvement 3: divapprox

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$

  Let $s \leftarrow m - n + 1$
- Compute:
  - Approx. quotient $Q^\star =< q_{s-1}, \ldots, q_1, q_0 >$
  - $A \leftarrow A - \sum_{i+j=n-2}^{n+s-2} q_i d_j B^{n-3}$

Improvement 3: divapprox

- Given:
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $D =< d_{n-1}, \ldots, d_1, d_0 >$

  Let $s \leftarrow m - n + 1$
- Compute:
  - Approx. quotient $Q^\star =< q_{s-1}, \ldots, q_1, q_0 >$
  - $A \leftarrow A - \sum_{i+j=n-2}^{n+s-2} q_i d_j B^{n-3}$

  Only affects top $s + 2$ limbs of A

Exact MP quotient

- Can compute exact quotient Q from divapprox (same cost)

Exact MP quotient

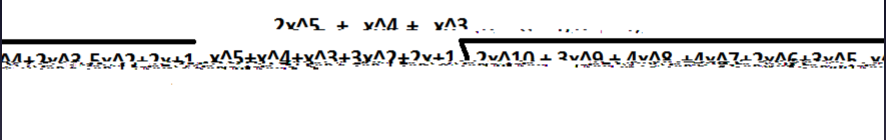
- Can compute exact quotient Q from divapprox (same cost)
- Can compute remainder R from Q using mullo

Exact MP quotient

- Can compute exact quotient Q from divapprox (same cost)
- Can compute remainder R from Q using mullo
- Complexity same as schoolbook divrem
- Can we compute integer divapprox in subquadratic time

Middle product

2004, Hanrot, Quercia, Zimmermann



Middle product

2004, Hanrot, Quercia, Zimmermann

$$
\begin{array}{r|l}
 & 2x^5 + x^4 + x^3 \\
\hline
x^5+x^4+x^3+3x^2+2x+1 & 2x^{10} + 3x^9 + 4x^8 + 4x^7+2x^6+3x^5 - x^4+2x^3-5x^2+2x+1 \\
 & 2x^{10} + 3x^9 + 4x^8 + 8x^7+8x^6+7x^5+3x^4+x^3
\end{array}
$$

Middle product

2004, Hanrot, Quercia, Zimmermann

$$
\begin{array}{r|l}
 & 2x^5 + x^4 + x^3 \\
\hline
x^5+x^4+x^3+3x^2+2x+1 & 2x^{10} + 3x^9 + 4x^8 \boxed{+4x^7+2x^6+3x^5} -x^4+2x^3-5x^2+2x+1 \\
 & 2x^{10} + 3x^9 + 4x^8 \boxed{+ 8x^7+8x^6+7x^5} +3x^4+x^3
\end{array}
$$

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?
- A. Yes. Relies on subquadratic integer mulmid

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?
- A. Yes. Relies on subquadratic integer mulmid
- (Harvey 2012) Karatsuba $\implies$ mulmid_toom42 (Tellegen's principle)

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?
- A. Yes. Relies on subquadratic integer mulmid
- (Harvey 2012) Karatsuba $\implies$ mulmid_toom42 (Tellegen's principle)
  - $A = < a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $B = < b_{n-2}, \ldots, b_1, b_0 >$

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?
- A. Yes. Relies on subquadratic integer mulmid
- (Harvey 2012) Karatsuba $\implies$ mulmid_toom42 (Tellegen's principle)
  - $A = < a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $B = < b_{n-2}, \ldots, b_1, b_0 >$
- $\mathrm{Mulmid}_{m,n}(A, B) = < ov_1, ov_0, p_{m-n}, \ldots, p_1, p_0 > = \sum_{i+j=n-1}^{m-1} a_i b_j B^{i+j-n+1}$

Subquadratic divapprox

- Q. Can we compute divapprox in subquadratic time?
- A. Yes. Relies on subquadratic integer mulmid
- (Harvey 2012) Karatsuba $\implies$ mulmid_toom42 (Tellegen's principle)
  - $A =< a_{m-1}, a_{m-2}, a_{m-3}, \ldots, a_1, a_0 >$
  - $B =< b_{n-2}, \ldots, b_1, b_0 >$
- $\text{Mulmid}_{m,n}(A, B) =< ov_1, ov_0, p_{m-n}, \ldots, p_1, p_0 >= \sum_{i+j=n-1}^{m-1} a_i b_j B^{i+j-n+1}$
- Roughly : computes middle third of $2n \times n$ product

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$
- Recursively compute top sh limbs of $Q^\star$ using DC divapprox

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$
- Recursively compute top sh limbs of $Q^\star$ using DC divapprox
- – Affects only top $sh + 2$ limbs of A

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$
- Recursively compute top sh limbs of $Q^\star$ using DC divapprox
- – Affects only top $sh + 2$ limbs of A
- Subtract Mulmid($Q^\star, D$) to adjust next sl limbs of A

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$
- Recursively compute top sh limbs of $Q^\star$ using DC divapprox
- – Affects only top $sh + 2$ limbs of A
- Subtract Mulmid($Q^\star, D$) to adjust next sl limbs of A
- – In total $(sh + 2) + sl = s + 2$ limbs of A affected

Divide-and-conquer divapprox

- $s \leftarrow m - n + 1$
- $sh = \lfloor s/2 \rfloor$, $sl = s - sh$
- Recursively compute top sh limbs of $Q^\star$ using DC divapprox
- – Affects only top $sh + 2$ limbs of A
- Subtract Mulmid($Q^\star, D$) to adjust next sl limbs of A
- – In total $(sh + 2) + sl = s + 2$ limbs of A affected
- Recursively compute low sl limbs of $Q^\star$ using DC divapprox

Problem

- Algorithm occasionally returns incorrect result!!

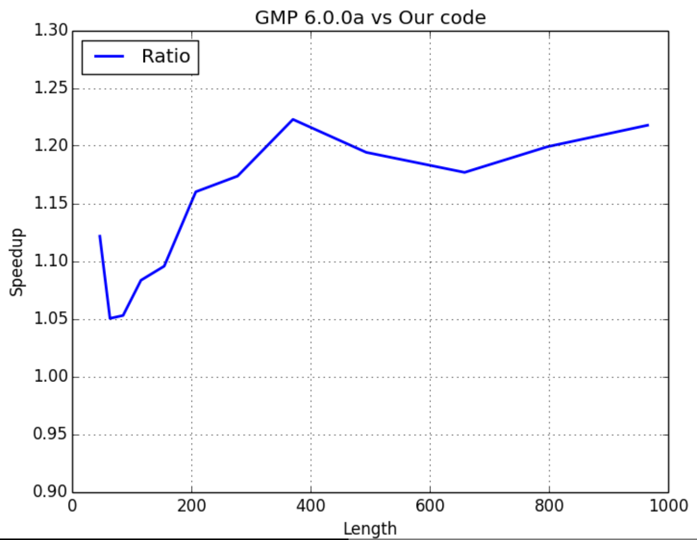
Problem

- Algorithm occasionally returns incorrect result!!
- Carries can cause each limb of quotient to be wrong

Problem

- Algorithm occasionally returns incorrect result!!
- Carries can cause each limb of quotient to be wrong
- Due to massive cancellation of correction terms, there is a linear time fixup to yield correct base B arithmetic

Timings (MP divappr)



Thank You

Thank You!