GMP: Fast and Accurate Bignum Arithmetic

by

Torbjörn Granlund, SWOX AB (tege@swox.se)

The GNU Multiple Precision Arithmetic Library (GMP)

- A library for arithmetic on arbitrarily large integers, rational numbers, and arbitrary precision floating-point numbers.

- Written to be portable...

- ...but faster than any other similar library!

GMP function overview

- About 120 integer functions (mpz_FUNC).
- About 20 rational number functions (mpq_FUNC).
- About 50 floating-point functions (mpf_FUNC).

Testing

Problem: Keep complex implementation of involved algorithms error-free.

Strategies

- Test vectors using special random numbers.
- Special "exhaustive" testing code for low-level vector functions (add, subtract, multiply-by-scalar, bit-shift)
- Defensive coding style for non time-critical parts.
- Simulating 4-bit computer allowing all bit patterns up to a certain size (nyi).

Portability

Written in portable C.

Hand-tuned assembly loops for a small set of critical operations:

- Bignum add, subtract, bitshift.
- Multiply of scalar and vector (three variants).

Assembly code is **optional**, thereby portability.

Performance

- Optimized for small operands (1-100 digits) by avoiding overhead.
- Optimized for average operands (100-10000 digits) by using good algorithms.
- Not yet optimized for huge operands. But work is under way!

What makes GMP fast?

- Low overhead for allocation, bookkeeping.
- Fast algorithms from literature.
- Careful implementation of algorithms.
- Hand-tuned assembly loops for all modern processors.
- Special functions for special operands (such as 2^n)

Why assembly?

- Assembly is hard to write and maintain...
- Assembly is unportable...
- Assembly becomes obsolete as new processors/processor implementations come to market.

  Sure, but:

- GMP gains 2-10 times speedup thanks to assembly code!

  This is because GMP needs aspects of the CPUs that are unavailable from a "high-level" language.

Assembly tuning techniques

- Loop unrolling.
- Software pipelining. "Juggle" data around around loop iterations.
- Special casing.
- Shallowing.

Example 1: Multiplication

New developments:

- Better Karatsuba implementation.
- Toom 3-way implementation.
- FFT.
- Benefits from low-level improvements (e.g. mpn_addmul_1).

Combined, these developments improve speed for all operand ranges.

Example 2: Division

New developments:

- Divide most significant words only and check remainder range.
- Burnikel/Ziegler for 2n/n.
- Benefits from low-level improvements (e.g. mpn_addmul_1).

Example 2: Division

[illustration not reproduced here]

All new developments in GMP

[list not reproduced here]

Acknowledgments

Important recent contributions by:

- Robert Harley (INRIA, Paris): Better Karatsuba and Toom 3-way

- Paul Zimmerman (INRIA, Lorraine): Burnikel/Ziegler division and new square root code.

The GNU Multiple Precision Arithmetic Library (GMP)

by

Torbjörn Granlund, SWOX (tege@swox.se)

GMP is available for anonymous ftp from
ftp://ftp.gnu.org/pub/gmp-2.0.2.tar.gz

GMP updates at the web page:
http://www.swox.com/gmp/