

Re: Salamin–Brent algorithm

Source: <http://sci.tech–archive.net/Archive/sci.math/2007–06/msg04313.html>

- *From:* Chip Eastham <hardmath@xxxxxxxx>
 - *Date:* Sat, 23 Jun 2007 16:37:32 –0000
-

On Jun 22, 1:40 am, dgoldsmith_89 <d.l.goldsm...@xxxxxxxx> wrote:

On Jun 21, 6:00 pm, Chip Eastham <hardm...@xxxxxxxx> wrote:

On Jun 21, 6:42 pm, Gerry Myerson <g...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

In article
<1182447514.837296.169...@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>,

dgoldsmith_89 <d.l.goldsm...@xxxxxxxx> wrote:

if I can just download say like
the first 2^{100} binary digits (or 16^{25} hex....

Um, 2^{100} digits – where would you put them all?
Seriously.

Anyway, no one has computed more than a few billion digits
(or bits, or whatever), which is way less than what you want.

Well, I agree with the thought (no one would have
room for 2^{100} bits), but actually the computation
of pi has been carried out to one trillion places
(10^{12}) in both decimal and hexadecimal arithmetic.

Re: Salamin–Brent algorithm

http://www.super-computing.org/pi_current.html

Thanks for the reference, that looks like it may be useful.

[snip]

Assume for the sake of argument that I want to use the S–B algorithm as described at

http://en.wikipedia.org/wiki/Gauss–Legendre_algorithm

this requires addition, subtraction, multiplication, division and root taking. Addition and subtraction aren't a big issue as far as affecting the number of digits one needs to keep around: they never change the maximum number by more than one, positive or negative.

Multiplication, division, and root-taking, however, are a different story: division and root-taking (multiplication when we're talking about the fraction-representing digits, which is of course almost all the digits in our calculation) right-shift the digits of the numerator/radicand, meaning that left-most digits which one might have thought one could dispose of because they represent precision achieved a while back in the calculation, may instead have to be kept around. But do they? If so, does this problem only delay one's ability to dispose of them, or must all digits always be used every step of the way (necessitating that one's storage capacity increase exponentially, since one's precision is so increasing with the S–B algorithm)? And if one can eventually dispose of left-most digits, what's the disposal criterion?

Does this make sense, or am I missing something which makes these questions nonsense?

DG

Re: Salamin–Brent algorithm

Re: Salamin–Brent algorithm

I think you are asking about generally how to implement multiple precision arithmetic. For the purpose of the quadratically convergent S–B algorithm to compute pi, a fixed–point (multiple precision) arithmetic package will do nicely.

Knuth's Art of Comp. Prog. vol. 2 (Semi–numerical algorithms) is a good foundational reference.

A simple answer to managing the extra digits produced in multiplication is to carry out the operation exactly, which may double the number of digits, and then round the result back to your "working precision".

Of perhaps greater importance is how to do the multiplication rapidly (a topic that Knuth devotes special attention to). Where addition and subtraction are easily seen to have implementations whose machine word operation counts are linear in the length of the operands (number of words in a multiple precision value), the common "by hand" algorithms for multiplication (and division) are going to require an operation count that grows like the square of the number of words in a value.

An immediate improvement on the $O(n^2)$ behavior just described is given by what is usually called Karatsuba multiplication, which in a basic form gives $O(n^{1.585})$ complexity [the exponent here being $\log(3)/\log(2)$].

The big gun for most multiple precision packages is to use Fast Fourier Transform methods to effect the product of two large integers in $O(n \log(n))$ complexity. Some links you may find interesting:

[Arbitrary precision computation
by Xavier Gourdon and Pascal Sebah]
<http://numbers.computation.free.fr/Constants/Algorithms/representation.html>

[Multiprecision benchmarks by D. J. Bernstein]
<http://cr.yp.to/speed/mult.html>

Re: Salamin–Brent algorithm

Re: Salamin–Brent algorithm

regards, chip

.