Gerald's Column
by Gerald Fitton

A new Internet List called Archive_42 was launched after I wrote last month's article. It is a spin off from Archive-on-Line and its 'Memoranda and Articles of Association' allow discussion of any subject except computers! Nowadays computers are so much a part of our daily lives that this restriction is difficult—particularly when you consider that every member of Archive_42 is an Archive subscriber!

There are topics which seem to be of interest to members which, on the face of it, have little to do with computers. Over the next few months I am going to risk discussing one such topic using as my flimsy excuse the fact that computers are used by practitioners in this field. Although much of what I have to say is really about the fundamental (and not so fundamental) background, I do promise you that somewhere (several somewheres) along the line I shall refer to the use of computers. What the computer does is to execute relevant sums with a precision and speed not available to an earlier generation.

This month I shall limit myself to a short introduction—but first, something else.

The Puzzle

In what follows the acronym FPE stands for Floating Point Emulator. This is a software module which executes all the arithmetic calculations in your machine. It is called an "Emulator" because the 'real thing' is a special hardware chip (such as a Maths Coprocessor) which was an option for earlier Archimedes machines. I wonder if any of you have used a machine which contains such hardware? Please let me know.

A couple of months ago I set a little puzzle. I said:

"I have always enjoyed Colin Singleton's Puzzle Corner so let me set one of my own. What is the largest prime number which can be represented by the FPE?"

As I write this (late January 2002), I have had only one reply. It is an extensive reply from a prestigious source; I shall share Colin Singleton's answer with you.

The Largest Integer

Of course there is no 'Largest Integer'. Even infinity can not claim this distinction so my puzzle refers to something a little simpler.

Colin says:

"The IEEE Double Precision Standard holds the mantissa of a floating point number in 53 bits (of which 52 are 'real' and 1 is implied) so it can hold numbers less than 2^53 with absolute precision. This standard is used by BASIC 64, SciCalc and all Spreadsheets."

So the first stage in solving my puzzle is to find the value of 2^53. If you have one of the newer Archimedes machines you will have more difficulty than Colin did using his A540 "Ancient machine". That is because the newer versions of the FPE will not return an accurate value if you enter the formula 2^52. My own experiments indicate that anything over 2^31 gives inaccurate answers with many of the more recent versions of the FPE. I believe that is because the ‘new’ method of finding 2^52 uses logarithms. I believe that the formula used is something like 2^52 = antilog (52*log 2). Whilst 2 and 52 are both Natural Numbers (and hence 2^52 must be a Natural Number) log 2 is not. Indeed, log 2 isn’t even a Rational Number; it is a Transcendental Number. No Transcendental Number can be stored accurately in a digital computer no matter how large the memory!

Here is an accurate answer. 2^53 is equal to 9 007 199 254 740 992. The largest integer which can be represented to the IEEE Double Precision is one less than this sixteen digit number. You can think of this number as being slightly smaller than 10^16.

Older versions of the FPE Acorn included a standard which they called Extended Double Precision (EDP); in EDP the mantissa contains 64 bits giving a precision of about 10^19. I shall play a little coy here and say that I don’t know whether that degree of precision is still available in the more recent versions of the FPE.

I shall play even more coy and quote Colin who says: “The FPE operates internally to a higher precision and rounds results to the level of precision specified by the application software.” I have seen a statement which means exactly the same as this in an early version of the Programmers Reference Manual (PRM). The statement in my (early) copy of the PRM says that the FPE uses a precision of 64 bits for internal calculations. This 64 bit precision is equivalent to about 10^19. Of course this is a greater precision than the 53 bits of the IEEE Double Precision Standard.

I shall leave it as an exercise to the reader to determine whether recent versions of the FPE do contain internal routines executed in Extended Double Precision.

The Answer

Colin has sent me his answer to my puzzle (backed up by some Basic PROCs and a FN). His number is definitely prime (I’ve checked it using both Fireworkz and PipeDream) and I am prepared to accept his assurance that there is no other prime number between the one he sent me and 2^53. I am not going to publish it yet but I will tell you a bit more about Colin’s method. It includes some techniques that you might find interesting—indeed, you might even find them useful.

Testing for Primality

Colin’s strategy is to start by checking the primality of (2^53 − 1). Then he counts down from there (in twos) checking for primality. His program stops at the first ‘winner’.

The only certain way of checking primality (whether a number is or is not prime) is to see if it is divisible by any prime number starting with 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, etc. There are quite a lot of prime numbers between zero and 2^53; the good news is that you don’t have to test every one of them. It is sufficient (and necessary) to check using primes between 2 and the square root af 2^53. If a number of the order of 2^53 is divisible by a number which is roughly its square root then the dividend will also be about the same size as the square root of 2^53.

How Many Primes

The square root of 2^53 is a bit less than 100 million.

Colin has a book which tells him that there are about 5.8 million prime numbers less than 100 million; he has checked this for himself (an exact answer) and the book is correct!

Finding these Primes

His first task was to find these 5.8 million prime numbers. He did this using a third century BC technique called “The Sieve of Eratosthenes”. Essentially this technique starts with all 100 million numbers in a file and then systematically ‘knocks out’ all those divisible by 2, then all those divisible by 3, then by 5, 7, 11, 13, 17, etc. All the numbers remaining after this operation must be prime.

The File of Primes

He has stored these primes in a single file which has a size of about 3.5 Mb. Those of you who are still with me will be wondering how it is possible to store 5.8 million (prime) numbers in a 3.5 Mb file. Colin has done so using a coding system which has a high compression ratio and is lossless (no loss of information). I found it most interesting.

What he did was to start with 100 million numbers and discard the 94 million or so of them which are not prime. Numbers divisible by 2, 3 and 5 are discarded neatly and efficiently by dedicated lines in his program. These lines are executed before using the ‘mod 30’ coding system described in the next paragraph.

Colin says (quite rightly) that when any prime number, n, greater than 7 is put into the formula n mod 30 then the remainder must be one of the following numbers: 1, 7, 11, 13, 17, 19, 23, 29. Because there are only eight possible values for the remainder (after division by 30) a single 8-bit byte can be used to indicate the primality of thirty consecutive numbers. This compression system reduces 30 numbers to one single byte!

If you consider that each of the 30 numbers might be represented by 64 bits (8 bytes) then this is a compression ratio of 30*8 = 240. Amazing!

Of course, this file of primes less than 100 million has to be constructed before trying to find the largest prime which is less than 2^53. This file of primes gradually builds ‘from the bottom up’ starting with 2, 3 and 5 (as special cases) followed by larger primes such as 7 and 29, before progressing to the ‘mod P’ technique which I shall describe below.

The ‘Mod P’ Technique

Colin much prefers to use Basic V rather than Basic 64—one reason is that Basic V is much faster. However, there is a problem with Basic V; the MOD function does not operate when the numbers are greater than 2^31. Colin overcomes this deficiency by using the rules of modulus arithmetic which I described in Archive October 2001.

Let me demonstrate with an example. To find out if a number such as 2^52 is divisible by a known prime which I shall call P, Colin first evaluates (2 MOD P), then doubles the answer and finds the remainder again using MOD P. This doubling process continues for 52 doublings (taking MOD P every time). At the end of this process the remainder will be zero if P divides exactly into the original number (2^52).

I'm sure that you will see how to extend this technique to numbers such as (2^52 + n) using the Addition Rule for modulus arithmetic which I described in an earlier article.

Colin's BASIC Program

Colin has not provided a complete program but some Basic PROCs and a FN which do the majority of the hard work. If you want to use his techniques then you will need to write a 'core' program which calls his PROCs and FN. Your 'core' can be just a few lines long.

Colin's files are on the Archive monthly disc.

If you study his PROCs you will see that he uses not one level of sieve as I have implied but a two level sieve. The first level is numbers up to 10201 and the second level extends just beyond the 100 million mark.

These 5.8 million primes are then used in a third level sieve which extends to 2^53.

I find his technique elegant.

The Solution

One reason why I am not giving you the solution is that I want to encourage you to construct your own BASIC program using Colin's PROCs and FN. If you follow Colin's advice then your program will run in Basic V never using a number greater than 2^31 to check the primality of numbers up to 2^53.

I have used Fireworkz and PipeDream to check that Colin's number is prime. The spreadsheets do run for a very long time so don't try to use them unless you are willing to leave your computer on all night! Basic is much faster.

An Anecdote

This month I shall provide only a 'trailer' to my chosen topic. The following story is something which really happened to me when I was teaching at the local College.

I tell my class that there is no such thing as a completely unbiased coin. I produce a coin. For those of you who remember them it was an old half crown made of sterling silver.

In a previous lesson I had spoken about 'fair odds'. I say that I am about to flip the coin and that it might come down 'Heads'. What are the 'fair odds' of it coming down 'Heads'? The 'correct' answer is that we don't know because, like all coins, this one must be slightly biased and we don't know by how much it is biased. So I'll accept an approximation.

The general consensus is that the probability of coming down ‘Heads’ is about 50%. It might be a bit different because of the inevitable bias—but 50% will be close.

I flipped the coin and it came down ‘Heads’.

I state that I am going to flip the coin again. What is the (approximate) probability of a ‘Head’ this time? Has it fallen because of the ‘Law of Averages’? Does it remain the same at 50%? Or, perversely, has the probability of it coming down ‘Heads’ increased in some mysterious and magical way?

This next bit really happened. I flipped the coin and it came down ‘Heads’ again.

I tell the class that I’m going to flip the coin again.

Same discussion about whether the probability of a ‘Head’ has remained the same (at 50%) or whether, because we’ve had two ‘Heads’ in a row, it must be even more likely that this next time it will come down ‘Tails’.

I am sure that all those of you who are into this kind of discussion with students will accept that it is more than a little difficult to persuade them all to agree on whether the probability of a ‘Head’ has fallen or stays the same (or even increased). In many ways Students are like ‘normal’ people and have their prejudices. The ‘Law of Averages’ is such a prejudice.

This is a true story. I flipped the pre 1920 half crown a third time and it came down ‘Heads’ again. If you have any feelings for me as a lecturer in front of all these very vocal students you will appreciate what a nightmare scenario was unfolding. Students are not totally ‘normal’ and I would like ‘on my side’.

Same question but we’ve now got more data about the coin. What is your best guess at the probability of the fourth throw returning the dreaded ‘Head’? You get a choice:

(a) The probability is still 50% (the coin doesn’t know what happened on previous throws)
(b) The probability is less than 50% (it’s about time our luck changed)
(c) The probability is greater than 50% (it’s got stuck in a rut).

Please write or email me with your thoughts about this at the address given in Paul’s Fact File. Please note that we no longer have a telephone or fax.

I did flip the coin for a fourth time—but you’ll have to wait until next month for the result.

The Maths Bit

If you have an unbiased coin (they don’t exist) then the chances of a single ‘Head’ is 1 in 2. The chances of three ‘Heads’ in a row is 1 in 8 (2*2*2). The chances of five ‘Heads’ in a row is one chance in thirty-two! Real coins (like my half crown) are never unbiased—so, for most real coins these figures are approximate.

We might have a ‘spreadsheet bit’ next month.