

Die kanadische Monatszeitschrift *The Computer Post*, Winnipeg veröffentlicht in mehreren Artikeln in ihrer Januar-Ausgabe 1995 eine exzellente erste Zusammenfassung des Debakels um den Defekt des Pentium-Mikroprozessors [1–2, 6] des Computergiganten Intel.

Im folgenden sowie in den wdv-notes Nr. 329 werden diese Artikel im Original nachgedruckt. Der Dank dafür geht an Sylvia Douglas von The Computer Post, 301 – 68 Higgins Avenue, Winnipeg, Manitoba, Canada, Email: *SDouglas@post.mb.ca*. Diese Artikel dürfen auch weiter-

verbreitet werden, wenn dabei die folgenden Spielregeln beachtet werden.

Permission is hereby granted to copy this article electronically or in any other form, provided it is reproduced without alteration, and you credit it to *The Computer Post*.

Do you need a replacement?

By SYLVIA DOUGLAS
The Computer Post

If you have a Pentium-based computer and you aren't sure whether you need a replacement chip, here are some points to think about. Risk is a product of two factors: how likely is it that the unpleasant event will happen, and how bad will the consequences be if it does happen?

In this case, the first factor is hard to assess. It isn't just that Intel and IBM disagree about the chances of an "average user" encountering an error. The really tough question is, how can you tell whether you are an average user?

As for the second factor – you are the best judge. You know what you are doing with your computer, who relies on the results, and how they might be affected if your numbers are wrong.

Do I have a flawed chip?

As of December 94, the answer is unequivocally yes. You do. All Pentiums currently on the market are affected. (Unless you are one of the select few who has asked for and already received a replacement.) Over the next few months, as corrected chips start to reach some manufacturers before others, the situation will become mixed.

*INTEL — The new standard
of accuracy.*

Read on Usenet [9].

The simplest way to identify a flawed Pentium is to start up the Windows Calculator program (one of the Accessories that comes standard with Microsoft Windows), and do one or both of the following tests:

- 1) Divide 5505001 by 294911
The Pentium's answer is: 18.666**00093**
The right answer is: 18.666**65197**
- 2) Divide 4195835 by 3145727
The Pentium's answer is: 1.333**73907**
The right answer is: 1.333**82045**

A 386, a 486, or a corrected Pentium (not to mention a Power Mac!) will give the right answer. The buggy Pentium gives the wrong answer. You can even write these numbers on the back of an envelope and try them in a computer retail store.

© 1995 – The Computer Post, Winnipeg

The test also works in Lotus 1-2-3 or Excel. Windows Calculator has accuracy problems of its own, and is not to be used for serious work, but it has been verified that for these two particular pairs of numbers, a wrong answer indicates the presence of the Pentium bug. At the very least, it is one program that is likely to be present even on a brand new machine.

If you are a programmer creating floating point programs which may run on a variety of machines, you can incorporate a test for the bug in your startup logic. A good example is p87test by Terje Mathisen [7]. This is a small assembly language program which will report several details about the processor and whether or not it has the bug. It is available on several FTP sites [7], including:

<ftp://math.ucdavis.edu/pub/fdiv/p87test.zip>
<ftp://www.isi.edu/pub/carlton/pentium/other/p87test.zip>

This will allow your program to decide at runtime whether or not to use the software fix currently being refined by Intel and a team of outside engineers.

Do my programs use floating point?

Some applications can be guaranteed safe; they don't use floating point arithmetic at all. For example, you are on solid ground if your Pentium is a standard network file server. E-mail handlers, communications software, database programs, general word processors – none of these are at risk. Most games like Doom use integer arithmetic for the sake of speed, and will not be affected.

Other programs such as home accounting programs may use floating point, may even do floating point divides, but not very intensively. The numbers they handle are usually small enough that a single divide error would not change the result the user sees on-screen.

Image processors, 3-D graphics programs, and some multimedia programs can make intensive use of floating point. However, see the section below on consequences; often, an error in this context will not make any practical difference. CAD software like AutoCAD uses floating point as well, and here an error may start to be serious.

In general, when you're dealing with shrink-wrapped software, be aware that there are calculation techniques which could result in magnifying the size of the Pentium error into something that would matter. Your program either does use such techniques, or it does

not... but since it is shrink-wrapped software, you don't know which alternative is true. You can call the software vendor's support line to ask them, but if they haven't specifically researched the question with a team of programmers looking hard at their code, even they may not be sure. (But they ought to be able to answer the easier question, whether the program uses floating point arithmetic or not. If it's doing only integer math, the program's immune to the bug, and the other question doesn't apply.)

*» You know postmodern –
Well, this is PostReality. «*

The Computer Post, Winnipeg.

The most common application that definitely uses floating point arithmetic is the spreadsheet. All spreadsheet programs come with libraries of software-based floating point routines, because they have to be able to run on older computers without hardware floating point units. Both Microsoft and Lotus have announced workarounds which will make their spreadsheets use these software libraries even on a Pentium, thus attaining accuracy at the expense of speed. Casual users may find this a satisfactory long-term solution, but to people who use complicated spreadsheets every day, performance matters. (Otherwise, why buy a Pentium?) For these users, software emulation can at best be a stopgap measure.

If you are writing your own programs, you can usually look at your code to see whether it uses floating point or not, whether it ever has occasion to divide, and whether iterative calculations are likely to magnify the effect of an error.

Programmers: Be aware that if you decide you do need a new chip, you should recompile all your programs once you get it, to be completely free of the risk. Some optimizing compilers pre-calculate certain values and embed them in the executable program. If one of these embedded values happened to be wrong because of the error, your program as compiled on the old system would still give wrong answers on the new one.

If you decide not to get a new chip, you will want to upgrade your compiler as soon as possible to one which incorporates Intel's software workaround. This will give accurate results even on a buggy Pentium by applying a scaling factor of 15/16 to a subset of divisions that are at risk of error [8]. The algorithm is presently being refined for efficiency, but it has already been proven to work in principle.

How about the work I've already done?

The only way to be absolutely sure it's right is to do it again on a different computer that does not have the bug, or on the same computer with a software fix installed. Obviously, you will only do this if the assured correctness of the results matters enough to be worth the effort.

But if it only happens once in 27,000 years...

There is a scale of risk, all the way from zero chance, to as frequently as once every millisecond if you're doing specific kinds of calculations. As Vaughan R. Pratt of Stanford University has pointed out, where a given user falls on this scale is very sensitive to tiny variations in the distribution of the data. Even one percent of the data clustering tightly near but not at the integers (in the so-called "bruised integer danger zone") can increase the chances of error by a factor of a million. The figure of 27,000 years is based on an assumption that data will be evenly distributed. The assumption may not be true. IBM has suggested that it is not the most plausible assumption [10].

Intel is now running large numbers of spreadsheet simulations at great speed on a supercomputer. They claim that their early results show the error situation is being encountered even less often than their previous estimate would suggest.

*INTEL = I Never Test
Enough Logic.*

Read on Usenet [9].

On the other hand, the only third-party study done so far, by PC Week Labs [3-4], indicates that the risk with their model of data distribution is about 200 times less than IBM's projections [10], but more than 200,000 times greater than Intel's estimates.

No matter what the outcome of this Battle of Probabilities, you don't really have any reason to care about the risk for the "average user", except as it reflects on the general credibility of the various parties. If such wide disagreement is possible, then the term "average" is meaningless. You want to know what your risk is, for your spreadsheets, and your other calculations. It may be vastly different from the average – much higher or much lower – because of subtle differences that are very easy to miss.

What are the consequences of an error?

In any program where floating point arithmetic is being used to calculate a dynamic screen display, you might see one pixel flash wrong for a fraction of a second. For static displays, the consequence might be one pixel that's permanently wrong, or it could be a whole picture that fails. One user quoted the example of FractINT, a fractal display program, which despite its name, does use floating point rather than integer arithmetic to calculate certain kinds of fractals. He had a set of numbers which had yielded a rather nice display on his 486, but turned into an ugly mess on his Pentium.

Not earthshakingly important, any of that. Going to the other extreme, some people are using their Pentiums to calculate wing-load-

ing for airplane design or radiation doses for cancer patients. They process medical images. They manage other people's money. They make major financial decisions for corporations. They run statistical analyses of experiments on the strength of which drugs are approved or rejected.

Here's the key point: If your Pentium is doing anything at all which would cause the word "liability" to cross your mind, for so much as a fleeting second, then you *do need an upgrade*, never mind the probabilities.

Even if your analysis (or your vendor's, or Intel's) convinces you that you are not running the slightest risk of incurring the error, you *still need the chip*. If you don't get one, you could find yourself in court some day trying to convince a non-technical judge and jury that continued use of a computer with a known bug did not constitute gross negligence. Describe the future courtroom scenario repeatedly in vivid detail to anyone who tries to talk you out of an upgrade, and you should have little difficulty in obtaining one.

[Ed. Note: This was written the day before Intel's change of heart [5], but it still applies to bosses, technicians, and other people who may say that an upgrade is too much trouble.]

What about the effect on resale value?

Computer capabilities and the demands of the latest software have been changing so fast that most people would rather buy new computers in the hope of staying current for at least a short while. Used computer prices today are already hard to estimate, never mind trying to guess three or four years into the future what the price effect of a bug will be. But yes, it seems reasonable that there would be some effect. How much, depends to some extent on how liberal Intel's chip replacement policy becomes as Pentiums grow cheaper, and the glare of publicity is removed from the issue. (These two influences may work in opposite directions.)

If Intel's replacement offer still stands, and can be transferred to a new owner, the effect on the resale price is likely to be limited to the cost (if any) of having the new chip installed.

Cost of installation?

Yes. Replacing a processor chip can be simple, or quite tricky, depending on the system design. You can do it yourself easily if: the chip socket is the Zero Insertion Force design; the heat sink just clips on, or your replacement comes with the right heat sink attached; and, there are no other components such as fans, disk drives, or CD-ROM drives placed so as to make it difficult to get at that part of the motherboard. Some computer designs use a special thermal-conductive grease between the heat sink and the chip. This is non-toxic, but messy. Oh, and don't forget to take precautions against an electrostatic discharge.

Even if it all looks straightforward, you may choose to pay a technician to install the new Pentium anyway, since applying power to an improperly installed chip can destroy it.

Intel, with possibly an ulterior motive, points out that the replacement procedure is more risky than their assessment of the effect of the error. Nevertheless, they are right in that if it is going to be done, it should be done care-

fully. [Ed. Note: Intel is now saying they will pick up the cost of installation at a contracted service centre.]

I don't want to justify my existence, I want a chip that works!

A lot of people agree with you. Intel is effectively being pressured into providing replacement on demand, although they are still trying to talk people out of wanting one. [Ed. Note: Well, they were.] Even if you "want it because you want it," you do have a right to get a product that works according to its specifications. Experience now seems to be that if you insist you want a new chip, Intel will give you one, but they still aren't making it easy. (Their policy may have changed by the time you read this.) [Ed. Note: It did [5].]

Remember that at this stage there are only so many new chips to go around. If after reading the above you feel you want a replacement, but your requirement is not urgent – there are other people who really do need one right now. Call Intel, by all means. Tell them that you have to have a corrected chip, but you can get by for a while, and you don't mind being put on a waiting list for 30 or 60 days. A scientist somewhere will thank you. □

Sonstige Hinweise

■ Die weltweite öffentliche Diskussion der Folgen des Pentium-Defekts wird im Internet vorwiegend in der Newsgruppe *comp.sys.intel* geführt.

Literatur

Alle im folgenden aufgezählten Materialien sind auch elektronisch publiziert und stehen als Dateien (Files) auf dem Ftp-Server *ftp.grumed.fu-berlin.de* im Verzeichnis *PC* zum Kopieren via Internet (*Anonymous Ftp*) zur Verfügung.

- [1] Dittberner, K.-H.: „Intel Pentium – The Chip that Redefines Mathematics“. FU Berlin (IfP): wdv-notes Nr. 307, 1994–1995.
- [2] Dittberner, K.-H.: Intel Pentium – Der eingebaute Divisionsfehler. FU Berlin (IfP): wdv-notes Nr. 327, 1994–1995.
- [3] What Does the Pentium Do, and When Does it Do It? – The „PC Week“ Labs White Paper, including source code for the float divide verification test suite in C. Published: 16. December 1994. – File: PBUG_PC-WEEK_PAPER.TXT.
- [4] A Question and Answer Paper Regarding the Pentium Flaw. Prepared by „PC Magazine“ and „PC Week“ Labs. Published: 16. December 1994. – File: PBUG_PC-MAGAZINE_QA.TXT.
- [5] Intel's Pentium replacement policy change. Press-release as found on their World Wide Web server on 20. December 1994. – File: INTELS_PRESSREL_2.TXT.
- [6] Smith, Richard M.: How Dr. Nicely's Pentium FDIV bug report reached the Internet (Early History). Published on UseNet's *comp.sys.intel* on 27. December 1994. – File: PBUG_HISTORY.TXT.
- [7] Mathisen, Terje: Test program for the Pentium chip FDIV bug. – File: P87TEST.ZIP (komprimierte Datei).
- [8] Dittberner, K.-H. (Ed.): Suggestions for Pentium Bug Software Workarounds. – File: PBUG_WORKAROUND.TXT.
- [9] The UseNet Community: A Intel/Pentium Jokes List. – File: PBUG_JOKES.TXT.
- [10] IBM's statement on their Pentium study. Published: 12. December 1994. – File: PBUG_IBM_STUDY.TXT.