

## Re: Disobeying jet engines – why?

---

*Source:* <http://sci.tech–archive.net/Archive/sci.electronics.design/2008–01/msg04998.html>

---

- *From:* Martin Brown <|||newspam|||@nezumi.demon.co.uk>
  - *Date:* Thu, 31 Jan 2008 09:43:57 +0000
- 

In message <13q1nrb5j31lc3f@xxxxxxxxxxxxxxxxxxxx>, Joel Koltner <zapwireDASHgroups@xxxxxxxx> writes

"Jan Panteltje" <pNaonStpealmtje@xxxxxxxx> wrote in message [news:fnqlse\\$ghu\\$1@xxxxxxxxxxxx](mailto:news:fnqlse$ghu$1@xxxxxxxxxxxx)

eh, using C, with the modern compilers highly optimising, they 'know' the processor', may actually improve performance, and reduce chances of errors that could have occurred by misunderstanding some CPU features  
by an asm programmer.

Something as simple as a standard CRC routine such as this one:

```
ULONG CalcCRC(ULONG rc,const UCHAR* buf,UINT len)
{
while (len > 0)
{
UCHAR nb = *buf;
UCHAR i = (rc & 0xff);
i ^= nb;
rc = rc >> 8;
rc ^= CRCTable [i];
buf++;
len--;
}

return rc;
}
```

...is a lot quicker to implement in C than it is in assembly; if I'm paying someone to do this C is certainly the way to go here. It's also then a lot faster to change the code around to use pointers (effectively changing the addressing mode used in the generated code) and check whether or not the performance improves (on this CPU it became a little worse).

This really isn't a particularly good example because on a lot of CPUs there is an almost 1 to 1

Re: Disobeying jet engines – why?

correspondence between each line of C here and an assembler instruction (or two). It is only a few lines longer in x86 ASM although somewhat more cryptic and decidedly non-portable..

A much more challenging test is to create a routine to transpose an NxN large matrix as quickly as possible. When the algorithm matters using a HLL is a big help. Or try to write a really strong chess program (hint: they are now almost all coded in HLLs with just tiny parts in assembler).

Similarly using the right language for the job in hand can make a huge difference to the amount of effort needed to complete it.

I challenge the ASM will do everything you need crowd to write a QUINE (a program which when executed will output itself in sourcecode form) in their favourite assembler language. I choose to do it in one line of LISP.

```
((lambda (x) (list x (list 'quote x))) '(lambda (x) (list x (list 'quote x))))
```

It is among the shortest and a very old language dating back to the 60's.

Much is 'vague', for example if you use FPGA, then you depend on the vendor's soft to create the interconnects from the HDL.

This is a good point that a lot of people overlook. Even if you're programming in assembly, you're still making use of the (seemingly reasonable, at first blush) assumption that the code will execute the way you wrote it. With superscalar CPUs that dynamically reorder program execution, that's a HUGE amount of complexity going on behind the scenes to give the \_appearance\_ that that's what's happening, and of course the end result is that pretty much all "big iron" (32 or 64 bit desktop) CPUs today have non-negligible errata lists of what doesn't quite work the way it should.

The errata lists are surprisingly short considering how much clever stuff like register colouring and speculative execution is going on the background to keep the thing fully utilised at every clock cycle.

I remember when Cyrix commissioned a full formal proof for the design of an 8087 compatible chip their resulting validation suite found around a couple of dozen previously unknown defects in the Intel chip.

Regards,

—

Martin Brown

—

Posted via a free Usenet account from <http://www.teranews.com>

.