

Re: a dozen cpu's on a chip

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2008-05/msg01678.html>

- *From:* Martin Brown <|||newspam|||@nezumi.demon.co.uk>
 - *Date:* Wed, 14 May 2008 16:40:12 +0100
-

MooseFET wrote:

On May 13, 10:19 am, Martin Brown <|||newspam...@xxxxxxxxxxxxxxxxxxxxxx>
wrote:

MooseFET wrote:

On May 13, 12:04 am, Martin Brown
<|||newspam...@xxxxxxxxxxxxxxxxxxxxxx>
wrote:

MooseFET wrote:

No nontrivial language can
ever be proven to be
impossible to crash.

Minimalist languages like Modula2 have
been proved to be formally
correct

I would like to see this proof. I seriously doubt that there isn't
an
error in it like the one pointed out by Godel in set theory. It
may
also not be a proof at all. Many so called proofs are just
arguments
that a proof could be done and not actual proofs. To be a
proof all
the steps must be shown.

Re: a dozen cpu's on a chip

It is the output of ISO WG13 so you have to buy the (very) expensive ISO standard if you want to play with it.

I am far too cheap to pay for it just to go looking for the mistake but I am going to assert without out proof that there certainly must be one. I assert this on the basis that humans don't do proofs and that computer checked proofs are only as good as the computer that checked it which its self must have been proven and so on "turtles all the way down".

It may have a mistake in it. I am no expert on WG13s work, but the premise of the verification checkers is that at the lowest level the checker checks itself and is checked by other independently developed tools. It is always possible that they all share some common design flaw, but the bootstrap process starting from a checker that is small enough for a manual proof would seem to be watertight if done carefully. Machine proofs of the 4-colour map theorem have exploited this methodology for instance.

Remember that hardware designs are generally designed and simulated on software tools before they are committed to fabrication.

However, there are several compilers still available which implement ISO M2. Personally I don't like their choice of syntactic sugar, but I can see why they did things. I subscribed to another M2 dialect. By the time the full spec was published most industrial people had moved to other languages for purely commercial reasons.

I had heard it was "practical" reasons. They hed to get product out the door and thus had to pick a language and go with it.

No. We got fed up waiting for the language committee to finish diddling with things and sloppy C / Windows GUI bindings eventually became a show stopper on the ubiquitous PC.

A strongly typed language was extremely painful to use with early windows bindings where everything was a more or less amorphous blob of 2 or 4 bytes coerced into random interpretations at the whim of the GUI designer and whatever he was smoking at the time.

The specification is in VDM-SL which can be checked by software tools.

Sly Fox Chicken Coop Gaurd Inc" at your service. The checking tools would have had to have been checked.

Indeed and my understanding is that they were. And the final tools were bootstrapped up from a trusted

Re: a dozen cpu's on a chip

Re: a dozen cpu's on a chip

simple tool that was manually proved by a team of mathematicians. Again you cannot absolutely rule out human error, but the approach is about the most robust anyone has come up with so far. Too few people can read these formal verifiable specification languages.

It isn't very easy to read though. A small piece is online as an example and is a bit out of date but gives the flavour.

<http://freepages.modula2.org/downloads/process2.pdf>

It doesn't state what happens if the extra add of the increment would overflow. Assume an 8 bit unsigned V

for V:= 1 to 245 by 70

you get loops for V=1, 71, 141, 211 and then what? Does it throw and overflow or not?

An interesting question and one that is not covered in this spec snippet (which I agree is a defect).

[...]

If you include hung systems in crashed ones, the problem becomes much harder (even though it started out as impossible).

Halting problems are notoriously difficult.

Yes as in impossible with a tool that checks in a finite time. There is always some bizarre sort of looping that will hang the checking tool or be missed.

But there is no reason not to use the static testing tools that are available for your chosen language. And also complexity measurements give a strong hint on where to look for coding errors.

Ada's high integrity subset for safety critical work is another one.

I think it was John Barnes at Praxis that did the high integrity stuff.

BTW: The ADA folks said "there shall be no subsets" early on.

Re: a dozen cpu's on a chip

Some Ada folks made subsets non-the-less. It was just too big.

I suspect LISP comes pretty close to being impossible to crash, but when you include practical implementations some of the OS impurities will provide ample opportunity to bring the thing to its knees.

Since LISP isn't compiled it is a lot harder to imagine it killing a system its self but the base interpreter can't be written in LISP so the error would be there.

It is a common misconception that Lisp is always interpreted. I was involved long ago in the development of a Common Lisp Compiler.

Interesting. "Lots of Inconvenient Single Parenthesis" has come a long way.

I prefer "Irritating". At the time LISP was in one of its periodic ascendencies as the most promising AI language and just becoming available on workstations. I expect it will do so again before too long.

When a pass by reference is done, the reference could contain the permissions and size information so that the called routine would be restricted to only writing where the callee wants it to.

You do have to be careful you don't create something that is unweildly. x86 has an array BOUND instruction but I can't recall ever seeing it used in anger.

Doing it at the hardware level isn't all that hard. The reference only needs to contain the starting address and the length in special hardware. Accesses would always use the reference and an offset into it. The hardware would have to have an adder and comparison circuit special to the addressing process.

Re: a dozen cpu's on a chip

Re: a dozen cpu's on a chip

The BOUND instruction already exists, but isn't often used.

Regards,
Martin Brown

** Posted from <http://www.teranews.com> **

.