An alternative BASIC for the Commander X16

Paul Robson (24 Oct 2023)

Introduction

This is an entirely new approach. Well, it isn't, but it's something that occurred to me last night when I got woken up by a Muntjac at 3AM.

One of the problems of having an interpreter and a compiler for the same language is keeping them in sync. Even with one developer coding both this can be something of a challenge. So why not get rid of the interpreter ?

The machine on the right is very obscure ; it's a Belgian machine called the DAI. (Though the design is entirely British).

It has a small but dedicated community, they are incredibly rare, and most people don't remember it.

However, it has a commonality with two other machines (the Swedish ABC-80 and the Russian BK-0010) in that the BASIC is a compiler. There is no interpreter per se other than the runtime.

I have seen this done for other machines after the fact (there's one for the Apple II) but these machines shipped with it.

Apparently (I've never seen one let alone used one) you can't actually tell the difference. You can do all the usual things you can do in Microsoft BASIC, type commands from the command line and run things interactively and so on.

Because the design is British, the software is all in English and the entire source code is available for the machine. I've been reading it this morning. (Which is sad ..). I can see how it's doing it, it's not massively different to Blitz.

Proposal

So the proposal is we (well I) do something like this for the X16.

Effectively the "interpreter" Blitzes the code on demand and runs that.

For technical reasons I haven't thoroughly thought through yet this may limit the structured coding a bit. I'm not sure WHILE loops will work. IF THEN ELSE multiline might be difficult.

But I'd base it on MS Basic with additions – most obviously long identifiers and procedures and locals. There'd be some changes but it would be close enough to MS Basic to make transfer not too onerous. Mostly removal of problematic things like the stack unwinding issues.

It fits the X16 well because even the base machine has a lot of memory (256k or 512k, can't remember) by 64k standards. It will mostly use the 8k pages for storage of code. Blitz experience tells me that tokenised BASIC code is 30-50% bigger than the Blitz P-Code, so I don't foresee any issues there.

It may be possible to implement something I've only seen (I think) on STOS on the Atari ST which is a task switching BASIC – it's hand task switched, not pretend multiprocessing. STOS used this very well, so you could run a sprite editor in one block of memory, a music editor in another and the BASIC source in another and flip between them with the function keys.

Adding the feature so it can take code from the editor is relatively trivial. I'd still keep the line numbers, but apart from compatibility features GOTO/GOSUB etc. and editing they wouldn't play a part.

It may be possible to make it talk to the BASIC transpiler as well, don't know enough about it.

As an example of potential gains, the DAI itself, which has a 2Mhz 8080 , has a BASIC noticeably faster than the C64.

As a rough comparator an 8080 at 2Mhz comes in at 0.145 MIPS. A 1Mhz 6502 comes in at 0.43 MIPS (apparently), so a C64 is about 3 times faster (and an X16 about 24 times faster) in terms of raw processing speed.

Rugg/Feldman Benchmarks

Benchmark	DAI	C64
1	0.93	1.2
2	4.78	9.3
3	10.06	17.6
4	9.78	19.5
5	11.20	21
6	18.12	29.5
7	30.11	47.5

These show the DAI to be somewhere between 1.5-2 times quicker than the C64 in operation. Scaling it up for the processor speed differences, this suggests it's 5-6 times quicker than the C64.

This speed up is roughly comparable for the non floating point improvements for Blitz (an 8080 will do better because it has 16 bit operations).

As far as I can make out, speaking not a word of Swedish or Russian, similar improvements occur in their machine's BASIC and I think they do much the same thing.

Downsides

Apart from memory, which I think is not a problem, the only slight downside would be a slower startup of a program. DAI compiles as you enter the code.

Partly because of the Editor option, and partly for speed reasons, I'd suggest compiling lines on demand as an approach. This will cause a slightly slower startup (you've doubtless noticed Blitz isn't instant, though it isn't slow either) as lines would be compiled on being encountered for the first time, however once compiled they would run flat out.

Compiled code wouldn't be saved with the source (though it might be feasible).

Next steps

Well have a read and let me know what you think (this doesn't preclude the maintenance of Blitz). Feel free to show it other people if you want.

Then an initial specification of the language *core*. No actual I/O functionality or Graphics or anything like that, just enough of the core working to check it is theoretically and useable and see what speed we can get out of the bits we can improve.

Then add the operators and commands, to suit.

(This is actually not difficult, adding commands like TILE takes very little time. In Blitz all the functions which talk to the sound chip are generated automatically , compiler and runtime)

Paul Robson

24 Oct 2023