Integrating Blitz into the X16 Hardware

Paul Robson 8th Oct 2023

Introduction

Blitz consists of two phases.

Compilation

Blitz can compile from any source that can return the tokenised lines of the program.

Compilation has to go into memory somewhere because there is a second pass of the object code to patch up all the line number references, as these can be forward. This is not a technical issue as the 512k RAM space can be used for this.

Besides this, memory is required for the compilation tables, of which there are two ; a mapping of line numbers in BASIC to object code addresses and a list of declared variables and their locations.

Runtime

Runtime code and data has to be held in RAM memory. It should be possible to put the p-code in the paged memory but this has not yet been tried. Unused memory is allocated initially for the variables, and then for things like arrays ; these are declared dynamically rather than at compile time in the original Blitz, which would not allow code like *70 dim bx(n):dim by(n):dim bc(n)* from balls.bas

Short Term Options

In many areas it's not clear how things actually work. The Compiler and Runtime would fit into ROM comfortably, and the code is designed to be ROMmable.

However, I don't know how this integrates with the Kernal, if at all. The Kernal could be like the BBC Micro sideways ROM system (very similar mapping of multiple images) in which it had a *<command> (using BASIC tokens for this is not ideal) and any commands it didn't understand were passed to plugin ROMs using the API.

This limits what can be done in the short term, e.g. the next week. It would be good, and useful for testing , if developers could download and play with it from your demonstration.

This doesn't limit any further ideas, like having a BLITZ command. The I/O and configuration of the compiler and runtime have been extracted in the same way as x16-assembler to make it fairly flexible about what it can use as source and object.

BASIC, P-Code Compiler, Runtime in RAM

One solution is to squeeze everything into the RAM memory. So the developer could edit in BASIC as they normally would, and run the compiler directly, which would compile the code, and if that was successful run it. This would be a neat demonstration but limited by memory (but Blitz really needs to be tested on smaller programs first anyway).

The downside of this approach is that you'd have to invoke blitz using SYS 20480 or similar to make it compile and run.

BASIC in file, Compiler/Runtime as seperate program, P-Code memory or as an executable file

This is closer to the way the original Blitz works ; you save your program on storage, run the compiler and it produces a runtime ; this could either be run automatically straight off or saved to disk (or both, I suppose !).

The workflow would be something like.

1. SAVE "SOURCE.PRG" – save current source
2. LOAD "BLITZCOM.PRG" – load (and then run) Blitz compiler, which loads in SOURCE.PRG and compiles it.
3. RUN
4. LOAD "OBJECT.PRG" – load the result in and run it.
5. RUN

So slightly longer.

This relies on the source code being called SOURCE.PRG and it generating a final file OBJECT.PRG (which is the combined runtime and p-code and can be run as normal)

(can MS Basic do RUN "OBJECT.PRG" or CHAIN "OBJECT.PRG" ? I haven't programmed in it since I had an 8k PET !)

The main variant in this is to miss out steps 4 and 5 and have the compiler run the code after its finished compiling it.

It would also be possible to have programs with any name and add the source code program as a parameter (if possible) or type it in as with the original, which would drive me mad.

Paul Robson.