# TinyC-1

This page is about bootstrapping TinyC-1, a typical bootstrapping C compiler, in assembly language, on your own custom architecture – in this case, the SD-8516

## The Plan

Well, I wrote the SD-8516, I spent a long time trying to implement BASIC on it, and then I did Forth. Forth was a lot easier than BASIC. In any case I learned quite a bit, and I think i'm up to the task of tackling C now. Pound for pound, it is not terribly difficult as long as you start with some very basic goals.

```
int main(void) { return 0; }
```

The first goal is simply to compile int main(void) { return 0; } and iterate from there. This at first glance seems simple, but if you attempt to do it “seriously” it may suddenly seem not-so-simple. But, it is. You just have to lay it all out. Right?

- 1. Basic REPL (interpreter prompt).
  - There's no filesystem or shell, so this will be a REPL like FORTH's. I'm removing FORTH for the development of this project.
- 2. Paste-in editing
  - The autotyper is the easiest way to get source into the system right now. It will take what's typed in the REPL and paste it against what it already has. We can do OPFS later maybe.
- 3. The Lexer
  - The Lex Luthor to my Superman. The hardest part. Can we do this token by token? Do we need to build an AST? I don't even know what that is right now.
- 4. Parser
  - According to my research I need a parser.
- 5. Emitter
  - Like Forth, we have to emit code. It shouldn't be too far off from emitting Forth words.

### The REPL

Based on BASIC's. Its not intended to be a language. This way is very easy for me to add new commands.