Programming in C for the SD-8516

Self-hosted TinyC v1

Yes, you can write C code for the SD-8516. However there are certain… limitations of the self-hosted project. Staying positive however, you can do quite a bit.

TinyC V1 can do..	**Not available**
Builtins (putchar, getchar, halt, yield)	no enums
int and char* (8-bit access)	no inits i.e. no int a = 5;
Expressions with precedence	no typedef, no casting, no conversions
Local variables	no multiple return types (int only)
if/else/else if	no static, extern, register, volatile, const
Comparison operators	no && or ||, no %, no bitwise ops
while	int *p; p + 1 adds 1 byte, not 3
break/continue	no sizeof
Multiple functions + function calls	no function pointers
Function arguments	no varargs
Pointers (read, write, address-of)	no void type
Array indexing	no int a[10]; pointers only
String literals	no type checking
Hex literals	char arithmetic doesn't promote to int
Character literals	no standard library
Comments (*)	no dynamic memory
Global variables	no file IO and no access to system INT libraries

What this is really is more of a baseline C you could write a better compiler in. That's where we are right now. Secondly, this is not packaged anywhere yet; I don't know if I want to release it because it is really not a very robust program; however, I very likely will, sometime in the fall.

Self Hosted TinyC v2

v2 is v1 written in C. It compiles but there are issues. The issues revolve around the fact that this type of computer system is not really intended to host and run a C compiler. It can be done but as I have learned, the easy way forward is to rewrite the memory map and the kernal. This creates a chicken and egg problem. I need to rewrite the kernal, but I don't feel the need to rewrite the kernal in C just to rewrite the C compiler.

v3 (LLVM Back-end)

The LLVM back-end is not self hosted, obviously. However, I am learning a lot about C by doing this and I may be able to use what I know to help write v4 under the v2 compiler. Or at least, write v4 in

v3 then try to back-work v1/v2 to compile that. Either way, this is the path forward.

What about right now?

Given all this I recommend you program in assembly first. That's really what this project is about, the retro way. You should give it a shot, if you've never tried assembly before. This platform is designed with hand-assembly in mind. Like the old days.

However, if you really want to program in C, well, talk to me, and maybe we can work something out where I try to package this for you to test it out.