A Note on Parsing Source Code

Corrolary to "Poor Man's Recognizer"

I pondered some more on the relationship between gforth's `[`, `]`, and `parser`.

When the Forth system is used for cross compilation, we must be able to switch back and forth between interpreting and compiling using `[` and `]` - just the same as in a normal host only system. But the `parser` action will be different depending on whether we want to produce code for the host or for the target system.

In gforth, `parser` is deferred, which is already a good starting point for differences in host or target code production, and `[` and `]` are defined as follows:

```
: ]  ( -- )   ['] compiler is parser   1 state ! ;
: [  ( -- )   ['] interpreter is parser   0 state ! ;
```

Bad luck if I have to modify `parser` depending on my cross compiler's needs, because `[` and `]` will overwrite whatever I may have assigned to `parser`. Actually, I invented this almost 40 years ago in volksForth. It seemed a good idea at the time but it was clearly a mistake.

This was one of my bad ideas, which I

A more useful implementation will look like this:

```
: ]  ( -- )   1 state ! ;
: [  ( -- )   0 state ! ;
: host-parser  ( c_addr u -- )
   state @ IF  compiler  ELSE  interpreter  THEN ;
' host-parser IS parser
```

8-Sep-2020, Klaus Schleisiek