http://www.forth-ev.de/wiki/doku.php/words:kernel_embedded:minimum_word_set

Minimum Word Set

How many primitives does a Forth need to get it started?
Surprisingly few, as you can see with Gforth EC, which provides a high-level definition for almost every primitive.

But beware! If you omit too much, you risk circular behaviour (loops).

We need in any case:

:dodoes	as generalized entry point for all high-level definitions, or	1
	Call for primitive-centric implementations.	
@	and	2
!	to access the memory.	3
>R	and	4
R>	for the return stack, so anything can be moved.	5
+	or	6
	2* for Artihmetik. And	
NAND	as a universal bit instruction.	7
?BRANCH	or	8
	0= for branches.	
	And finally	
;S	and	9
EXECUTE	for execution.	10

Everything else can be defined from these words,
even if it then needs temporary variables:

```
: lit'    r> @ ;

: cell    lit' [ 2 , ] ;    \ oder 4 oder 8 ...

: tmp1    lit' [ here cell+ , 0 , ] ;

: dup     tmp1 ! tmp1 @ tmp1 @ ;

: lit     r> dup cell + >r @ ;
```

By Bernd Paysan

Translation Juergen Pintaske 2015_11_21