Subject: faster forth
Posted by blk on Thu, 30 May 2013 03:58:04 GMT
View Forum Message <> Reply to Message

Message-ID:
Date: Fri, 12-Oct-84 13:10:58 EDT
Article-I.D.: syteka.491
Posted: Fri Oct 12 13:10:58 1984
Date-Received: Sun, 14-Oct-84 06:51:39 EDT
References: iham1.233
Lines: 44

The BYTE article referred to
Re: FORTH returns on a TI 99/4A
is apparently correct in the discussion of speed versus size.

Note that the size increase might be considerable, as the body of every
high level definition becomes half again as large (+50%)!

However, there is no need to rewrite FORTH to achieve the speed increase.

The author of the BYTE article made some theoretical errors that cloud
the issue a bit. The change does not affect the dictionary structure in
any way, so the new fastwords are compatible with existing FORTHs.
Changing from pointers to calls in a definition transforms the
fastwords from a Threaded Interpretive Language (TIL) construct
into what might be called a Threaded Hybrid Language (THL) construct.

I choose the word 'Hybrid' because the fastwords can co-exist with
regular forth words. The BYTE article referred to a language implemented
this way as another form of a TIL, but in fact words following this
form are directly executed, not interpretted.

To use fastwords, one only has to create a new defining word,
maybe called fast: . This word will work just like : except
that it will compile calls to other fastwords, not pointers to
words as : does, and the
code field in the new word points to the parameter field like
the normal forth primitives.

The easiest way to hook it all together is to invoke fastwords with
a 'run' command or some such; somehow you need to return to the
inner interpretter when the fastwords are done. However, tricky things
can be done with the fast: defining word to make fastwords look
and act the same. Similarly, one can call regular TIL forth words
from a fastword with a clever fast: defining word. You just have

to trick ;S into returning to the fastword instead of the interpretter.

Hope that this helped.

ps- if the parent forth doesn't use the machine's return stack as
    the FORTH return stack, the compatibility is lessened.
    My favorite FORTH doesn't, but most do.

B<