EuroForth 2016, Reichenau/Konstanz

U. Hoffmann <uh@fh-wedel.de>
FH Wedel

Implementing the Forth Inner Interpreter in High Level Forth

Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Overview

Introduction

Threaded Code

- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation

- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Introduction

- development of hard real time system
- desire to have an interactive command shell while still meeting the real time requirements
- idea: have a controllable Forth interpreter (inner & outer) stepwise execution

Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

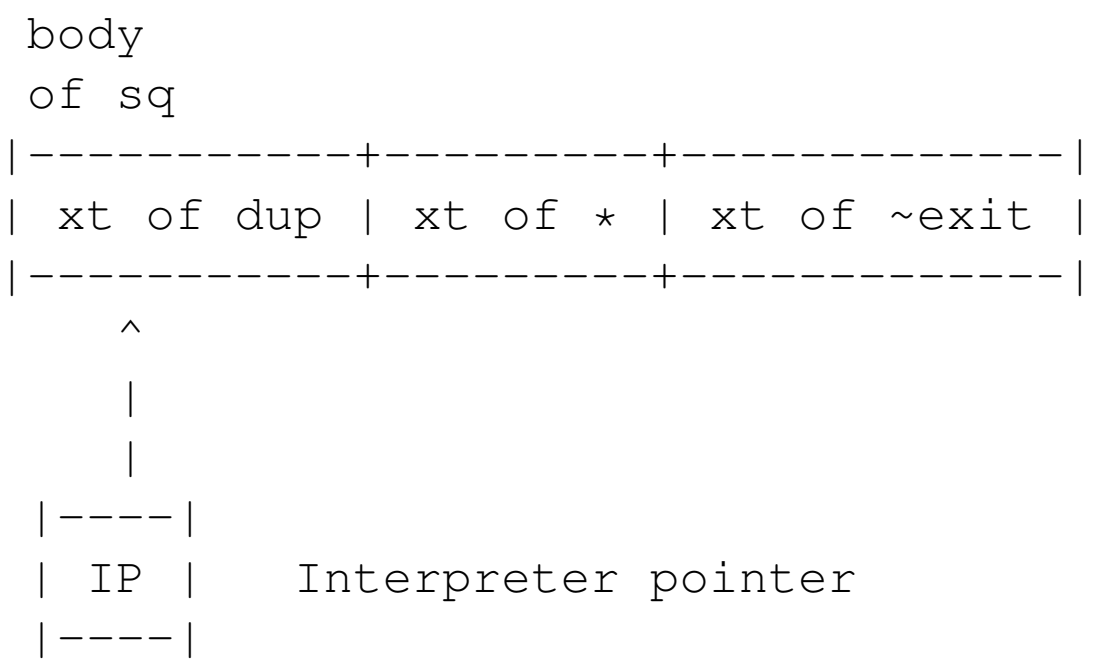
Conclusion

Demo

Threaded Code

- words of underlying system become primitives
- only threaded code no structure of header/dictionary

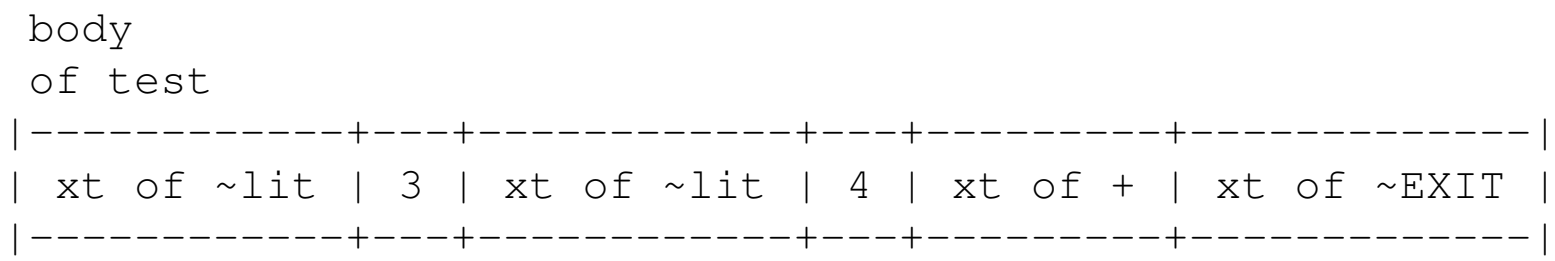
```
~: sq ( x – x ) dup * ~;
```



Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Number literals

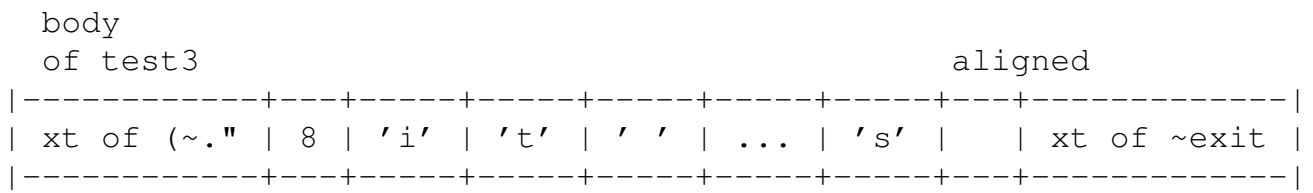
```
~: test ( - u ) 3 4 + ~;
```



Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Printing string literals

```
~: test3 ( – ) ~." it works" ~;
```



Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code

Number literals

Printing string literals

Control structures

Limitations

Implementation

Inner Interpreter

Return stack operations

Inline number literals

Printing inline string literals

Control structures

Compiler to threaded code

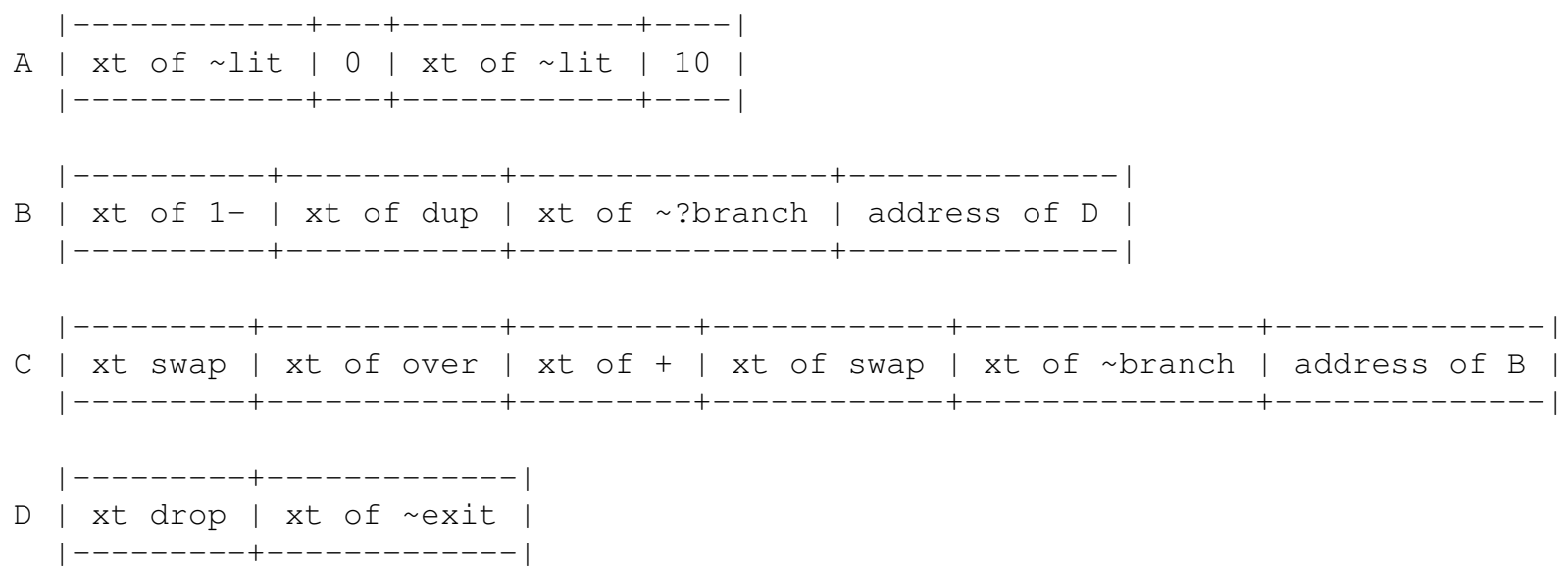
Interpreter in threaded code

Conclusion

Demo

Control structures

```
: looptest ( -- )
   0 10
   ~BEGIN
     1-  dup
   ~WHILE
     swap over + swap
   ~REPEAT drop  ~;
```



Limitations

- (user) variables or constants.
- `DO LOOP`s.
- Neither defining words nor `DOES>`
- a primitive for pushing address and length of string literals on the stack (like `S"`).

Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Implementation

- Inner Interpreter
- Return Stack
- Compiler *to* Threaded Code
- Outer Interpreter *in* Threaded Code

Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Inner Interpreter

```
Variable IP  0 IP !

\ Perform a single interpretation step
: step ( i*x -- j*x )
   IP @ dup cell+ IP !
   @ catch
   ?dup IF cr ." Error " . reset THEN ;

\ Loop steps
: run ( i*x -- j*x )
   BEGIN IP @ WHILE  step  REPEAT ;
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Interpreter State

- Interpreter Pointer `IP`:

```
Variable IP  0 IP !
```

- `~R0` with corresponding return stack pointer `RP`.

```
Create ~R0  20 cells allot
Variable RP  ~R0 RP !
```

- a data stack shared with the underlying system
- memory (code and data) also shared with the underlying system.

Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Return stack operations

```
Create ~R0  20 cells allot
Variable RP  ~R0 RP !
```

```
: ~>r ( x -- )
  \ push a cell to the return stack
  RP @ !   1 cells RP +! ;

: ~r> ( -- x )
  \ pop a cell from the return stack
  -1 cells RP +!   RP @ @ ;
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

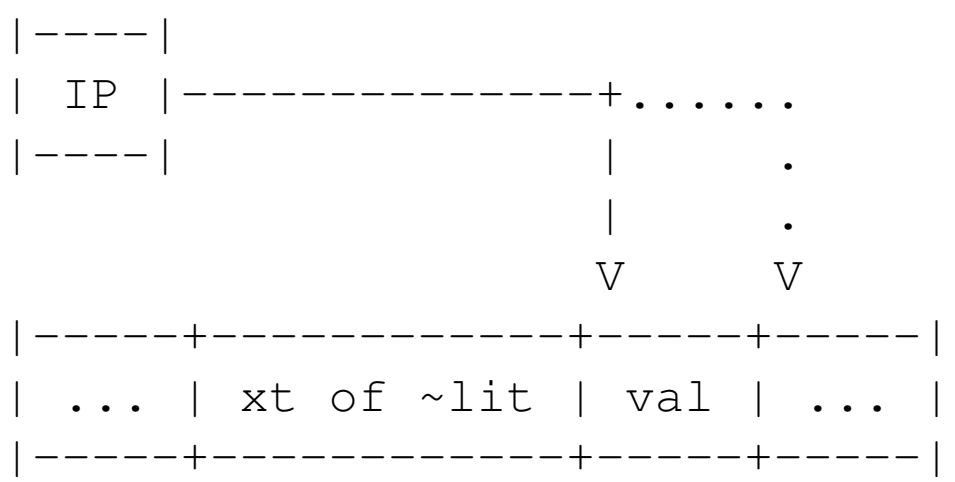
Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

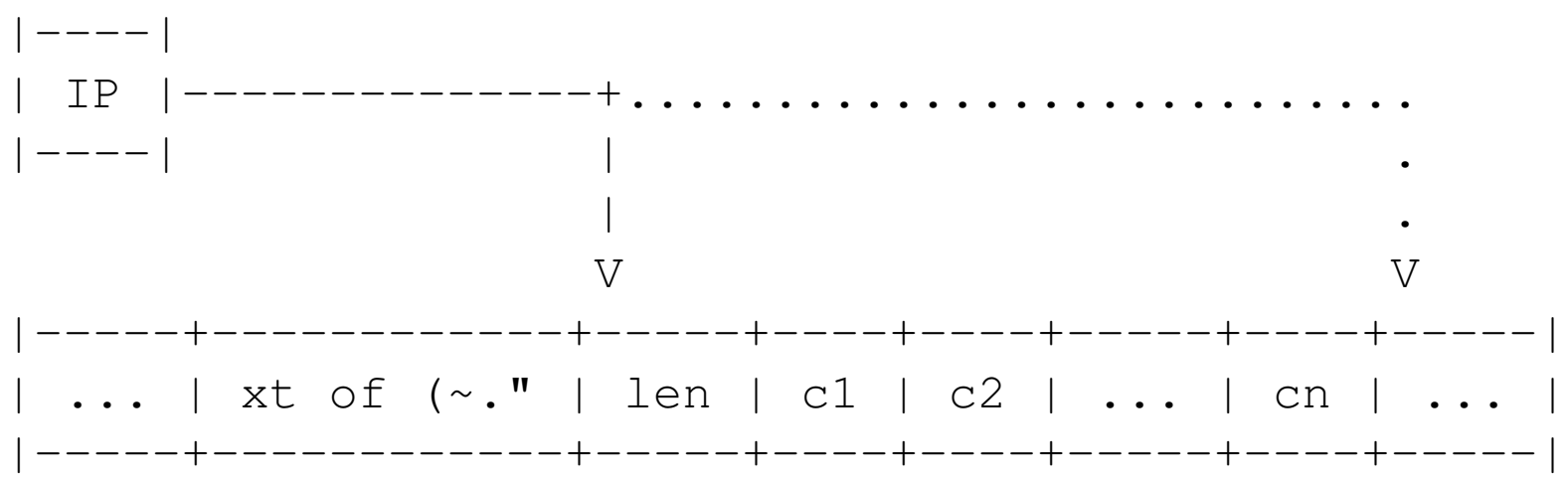
Inline number literals




```
: ~lit ( -- n )
  \ extract inline number literal
  IP @ @  1 cells IP +! ;
```


Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Printing inline string literals




```
: ~." ( <ccc>" -- )
  \ Compile inline string to be
  \ printed later when executed.
  \ Like ." but for threaded code
  ['] (~." ,
  [char] " word count
  here over 1+ chars allot place align
; immediate
```


Implementing the Forth Inner Interpreter in High Level Forth
Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo

Control structures

```
: ~branch ( -- )
   \ absolute unconditional jump
   IP @ @ IP ! ;

: ~?branch ( f -- )
   \ absolute conditional jump
   IF  1 cells IP +!
   ELSE ~branch THEN ;
```

```
: ~IF ( -- x )  ['] ~?branch , here 0 , ; immediate
: ~AHEAD ( -- x )  ['] ~branch here 0 , ; immediate
: ~ELSE ( x -- x' )  ['] ~branch , here 0 , swap here swap ! ; immediate
: ~THEN ( x -- )  here swap ! ; immediate

: ~BEGIN ( -- x ) here ; immediate
: ~WHILE ( x1 -- x2 x1 )  ['] ~?branch ,  here 0 , swap ; immediate
: ~AGAIN ( x -- )  ['] ~branch , , ; immediate
: ~UNTIL ( x -- )  ['] ~?branch , , ; immediate
: ~REPEAT ( x2 x1 -- )  postpone ~AGAIN  postpone ~THEN ; immediate
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Compiler to threaded code

```
variable ~state   ~state off  \ threaded code outer interpreter state

-13 Constant #notfound

: ~] ( -- )  ~state on
   BEGIN ( )
      BEGIN ( )
        bl word dup c@   \ scan next token
      WHILE ( c-addr )   \ another token found
        find  ?dup       \ look up in dictionary
        IF   -1 = IF ,  ELSE execute  THEN  ~state @ 0= IF EXIT THEN  \ found
        ELSE   0 0 rot count
          over c@ [CHAR] - = dup >r IF  1- swap char+ swap THEN \ word not found
          >number IF  #notfound throw THEN
          drop drop r> IF negate THEN
          ['] ~lit , ,  \ compile threaded code literal
        THEN
      REPEAT ( c-addr )  \ no more tokens in input stream
      DROP
      SOURCE-ID 0= IF CR ." ] " THEN
      REFILL 0=    \ read more from input stream
   UNTIL ;  \ input stream exhausted
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Compiler to threaded code

```
: ~[ ( -- )  ~state off ; immediate   \ stop threaded code compiler

: ~: ( <name> -- )
   \ push IP to return stack and set IP to start of threaded code.
   Create ~]  Does> IP @ ~>r  IP ! ;

: ~EXIT ( -- )
   \ Pop IP from return stack
   ~r> IP ! ;

: ~; ( -- )
   \ Compile end of definition and leave threaded code outer compiler
   ['] ~EXIT ,   ~state off   ; immediate
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

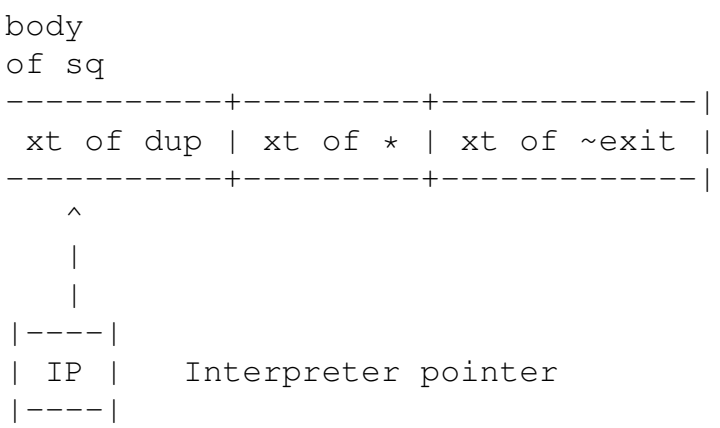
Demo

Compiler to threaded code

```
~: name ....    ~;

 name run
```

```
~: sq ( x – x ) dup * ~;
```



Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Interpreter in threaded code

```
~: ~interpret ( -- )
    ~BEGIN ( )
      bl word dup c@    \ scan next token
    ~WHILE ( c-addr )   \ another token found
      find              \ lookup in dictionary
      dup  1 = ~IF drop execute  ~ELSE   \ immediate
      dup -1 = ~IF drop state @ ~IF  compile,  ~ELSE  execute ~THEN ~ELSE
      \ word not found, number?
      drop   0 0 rot count   over c@ 45 = dup ~>r ~IF  1- swap char+ swap ~THEN
           >number ~IF  #notfound throw ~THEN drop drop    \ maybe number
           ~r> ~IF  negate ~THEN
           state @ ~IF  postpone LITERAL  ~THEN  \ compile literal
      ~THEN ~THEN
    ~REPEAT ( c-addr )
   drop ~;

   ~: ~quit ( -- ) clear-stack   ~R0 RP ! ~state off  interpret-mode
     ~BEGIN cr state @ ~IF ~." ] " ~THEN  ~query ~interpret ~." ~ok"  ~AGAIN ~;



~quit run
```


Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Conclusion

- threaded code structure for Forth colon definitions
- inner interpreter, `step`, in high level Forth.
- a compiler to generate this threaded code
- an interactive outer interpreter *in* threaded code controllable by `step`.
- interactive command interpreter in hard real time system

Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Demonstration

Demo

Implementing the Forth Inner Interpreter in High Level Forth

Introduction

Threaded Code
- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation
- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Summary

Introduction

Threaded Code

- Number literals
- Printing string literals
- Control structures
- Limitations

Implementation

- Inner Interpreter
- Return stack operations
- Inline number literals
- Printing inline string literals
- Control structures
- Compiler to threaded code
- Interpreter in threaded code

Conclusion

Demo

Introduction
Threaded Code
Number literals
Printing string literals
Control structures
Limitations
Implementation
Inner Interpreter
Return stack operations
Inline number literals
Printing inline string literals
Control structures
Compiler to threaded code
Interpreter in threaded code
Conclusion
Demo