Henceforth

Henry Hudson

November 8, 2024

Contents

1 About iii
- 1.1 links to source code . . . iii
- 1.2 Support . . . iii

2 Goals iv
- 2.1 Intergrate with Bitcoin (BSV) . . . iv
- 2.2 Get people coding . . . iv
- 2.3 Access to files app . . . iv
- 2.4 Further maths . . . iv
- 2.5 Settings . . . iv

3 Credits v
- 3.1 Starting Forth Leo Brodie . . . v
- 3.2 BITSRFR . . . v
- 3.3 Bitcoin . . . v

4 Documentation vi
- 4.1 Built in words . . . vi
  - 4.1.1 Compile words . . . vi
  - 4.1.2 Arithmetic Operators . . . vi
  - 4.1.3 Comparison Operators . . . viii
  - 4.1.4 Bitwise Operators . . . ix
  - 4.1.5 Stack Operators . . . x
  - 4.1.6 Return stack Operators . . . xi
  - 4.1.7 Decision Operators . . . xii
  - 4.1.8 Loop Operators . . . xii
  - 4.1.9 Base operations . . . xiii
  - 4.1.10 Pop up operations . . . xiv
  - 4.1.11 Constants and variables . . . xiv
  - 4.1.12 Terminal operators . . . xvi
  - 4.1.13 other words . . . xvii
  - 4.1.14 Networking operations . . . xviii
  - 4.1.15 OPCodes . . . xviii

Chapter 1

About

1.1 links to source code

github link
   Running project requires Xcode run on any iphone / ipad device will soon be optimized for all apple products.
   Private for the time being get in touch if you would like acess.

1.2 Support

Twitter
   Follow us on Twitter.

Chapter 2

Goals

2.1 Intergrate with Bitcoin (BSV)

(i) Fetch latest block headers

(ii) Intergrate with SwiftBSV for transactions, verification, wallet functions etc.

(iii) OPCodes file available to load and make use of

(iv) Make use of TimeStamp server for users words and CRDT (Conflict-free Replicated Data Type) system

(v) Full scrypt capability, each forth file will create a Scrypt that runs the op codes within

2.2 Get people coding

Provide a simple interface for FORTH programming language making it easy for people to learn to code.

2.3 Access to files app

If permission is granted by the user access to their documents will allow the program to load .fs .forth and possibly other files for the user to use how they wish.

Create read update delete CRUD operations should be supported, at least for FORTH files

if access is granted files will be stored on users icloud, for backup/saftey

2.4 Further maths

Eventually add functionality for trigonometry and calculus

2.5 Settings

Color customisation is available for users to pick a foreground color and a background color. also allow users to pick between the various keyboards.

Chapter 3

Credits

3.1 Starting Forth Leo Brodie

Leo Brodie Author of Starting forth a great text book, suitable for both novices and professionals.
    Starting FORTH book

3.2 BITSRFR

Fellow surfer of the binary ocean who has made FORTH tutorials and is making a swift command line FORTH application
   Youtube cruxFORTH github

3.3 Bitcoin

The greatest monetary system ever created. invented by Craig Wright aka Satoshi Nakamoto, please read the white paper.
   cruxFORTH github

Chapter 4

Documentation

Everything in FORTH is a word seperated by spaces. Thats all there is to the syntax. Its so simple.

FORTH is LIFO

FORTH is a stack based programming language which operates on a Last In First Out principle.

4.1 Built in words

Ideally we give the user just enough to make use of FORTH and through FORTH some of swifts functionality such as networking requests, the timer built into swift, etc. But not so much as to be relied upon, user creation of words should be encouraged and if possible a community data base of words and their definitions be made.

4.1.1 Compile words

One of the great thing about FORTH is its ability to extend the compiler during run time.

Compile Operators

:	puts the user into compile mode where they can start to define their own word.
;	exits compile mode so that word interpretation can again occur.
recurse	calls calls the word being executed.
exit	exits

4.1.2 Arithmetic Operators

Arithmetic is done using reverse polish notation, so the numbers are required on the stack before the artimetic operator can be used. This is done for clarity and efficency purposes.

Arithmetic Operators

`+` (n1 n2 -- sum)
Takes the top two items from the stack and returns their sum to the stack.

`-` (n1 n2 -- difference)
Takes the top two items from the stack and returns their difference to the stack.

`*` (n1 n2 -- product)
Takes the top two items from the stack and returns their product to the stack.

`/` (n1 n2 -- quotient)
Takes the top two items from the stack and returns their quotient to the stack.

`*/` (n1 n2 n3 -- result) result = (n1*n2/n3)
Takes the top three items from the stack the first two items are multiplied together, this result is then divided by the last item taken from the stack and the result appended to the stack.

`mod` (n1 n2 -- modulus)
Takes two items from the stack, divides the first by the second and the remainder of the division is added to the stack.

`/mod` (n1 n2 -- modulus quotient)
Takes the last two items from the stack and returns their modulus and their division to the stack.

`abs` (n - absolute value of n)
Takes the last value from the stack and returns its absolute value.

`negate` (n -- negated)
Takes the last item from the stack, and returns its value negated.

`max` (n1 n2 -- max)
Takes the last two values from the stack and returns the greatest value to the stack.

`min` (n1 n2 -- min)
Takes the last two values from the stack and returns the smallest value to the stack.

```
sumStack (n1 ... n -- sum of stacks values)
         adds all items on stack and appends the sum to the stack.
```

4.1.3 Comparison Operators

NOTE: f is flag or bool [true or false]
in Henceforth 1 is true and 0 is false

Comparison Operators

```
=    (n1 n2 -- f)
     Takes the top two values from the stack checks if they are
     equal, then returns the flag to the stack.

<>   (n1 n2 -- f)
     Takes the top two values from the stack checks if they are
     NOT equal, then returns the flag to the stack.

<    (n1 n2 -- f)
     Takes the top two values from the stack checks if n1 is less
     than n2, then returns the flag to the stack.

>    (n1 n2 -- f)
     Takes the top two values from the stack checks if n1 is
     greater than n2, then returns the flag to the stack.

and  (n1 n2 -- f)
     Takes the top two values from the stack checks if they are
     both the true flag, then returns the flag to the stack.

or   (n1 n2 -- f)
     Takes the top two values from the stack checks if either
     is a true flag, then returns the flag to the stack.

0=   (n1 -- f)
     Takes the top value from the stack, checks if it is equal to
     Zero, then returns the flag to the stack.

0<   (n1 -- f)
     Takes the top value from the stack, checks if it is greater
     than zero, then returns the flag to the stack.

0>   (n1 -- f)
     Takes the top value from the stack, checks if it is less
     than zero, then returns the flag to the stack.
```

4.1.4 Bitwise Operators

Bitwise Operators

```
and     (n1 n2 -- AND)
        Takes the top two values from the stack and returns the
        result of their logical AND to the stack

or      (n1 n2 -- OR)
        Takes the top two values from the stack and returns the
        result of their logical OR to the stack

xor     (n1 n2 -- XOR)
        Takes the top two values from the stack and returns the
        result of their logical XOR to the stack

invert  (n -- NOT)
        Takes the top value from the stack and returns the
        result of its logical NOT to the stack

rshift  (n1 n2 -- n1 bitwise right shift by n2 )
        Takes the top two values from the stack and returns the
        result of n1 biwise rightshifted by n2

lshift  (n1 n2 -- n1 bitwise left shift by n2 )
        Takes the top two values from the stack and returns the
        result of n1 biwise leftshifted by n2
```

4.1.5 Stack Operators

Stack Operators

dup	(n - n n) The top value is taken from the stack, duplicated and both the orignal and copy returned to the stack.
swap	(n1 n2 -- n2 n1) the last two values are taken from the stack, swapped and then returned to the stack.
2dup	(n1 n2 -- n1 n2 n1 n2) the top two values from the stack are taken, copied and both the original and copy returned to the stack.
over	(n1 n2 -- n1 n2 n1) The second value from the stack is copied and added to the stack.
rot	(n1 n2 n3 - n2 n3 n1) The top three values of the stack are taken and rotated left then returned to the stack.
-rot	(n1 n2 n3 - n3 n1 n2) The top three values of the stack are taken and rotated right then returned to the stack.
tuck	(n1 n2 -- n2 n1 n2) The top two values from the stack are taken. The top value is then copied and placed before the second value in the stack (third value deep)
2drop	(n1 n2 --) The top two values are taken from the stack and discarded.
2swap	(n1 n2 n3 n4 -- n3 n4 n1 n2) The top four values are taken from the stack, and the orders of the two pairs are reversed both as a pair of numbers and a pair of pairs and returned.
2over	(n1 n2 n3 n4 -- n1 n2 n3 n4 n1 n2) The first four values are taken from the stack, with the items 3 and 4 deep being copied and added to the stack.
reverseStack	(n1 n2 ... nlast -- nlast ... n2 n1) the first shall become last and the last shall

```
                become first. the stacks order is reversed

stackCount      ( -- n)
                The number of items on the stack is added to the
                stack.
```

4.1.6 Return stack Operators

Return stack Operators

```
>r   (n -- )
     takes top value from the parameter stack take moves it to the
     return stack.

r>   ( -- n)
     takes top value from the return stack and moves it to the
     parameter stack.

r@   ( -- n)
     copies last item from return stack.

i    ( -- n)
    copies top item in return stack.

i'   ( -- n)
     copies second item in return stack.

j    ( -- n)
     copies third item in return stack.

nth  ( -- n)
     copies the nth item in the return stack.
```

4.1.7 Decision Operators

NOTE: f is flag or bool [true or false]

Decision Operators

```
if     (f -- )
       Takes a flag from the stack and executes a words if true
       flag

else   (f -- )
       If flag taken from if was false these words will be exeuted
       This is optional

then   (f -- )
       Denotes the end of the if statement.

not    (f -- f)
       Takes a flag value and reverses its result.

within (n1 n2 n3 -- f)
       Takes three values from the stack and checks if n2 is a
       number within n1 and n3

true   (- true flag)
       Adds the true flag result to the stack.

false  (- false flag)
       Adds the false flag result to the stack.
```

4.1.8 Loop Operators

Loop Operators

```
do    (n1 n2 --)
      n1 is endValue n2 is startValue repeats words inbetween do
      and loop from startValue until endValue is met

loop  ( -- )
      Ends loop

+loop (n1 -- )
      loops in n1 increments.

every (n1 -- )
      Performs word after every, every n1 seconds.

stop  ( -- )
```

```
      Stops every running.

begin ( f -- )
      Marks the start of a while loop.

until ( -- )
      Ends while loop when condition is met.
```

4.1.9 Base operations

Much like FORTH heceforth accepts integers as binary, octal, decimal and hexidecimal. Just be sure to change the base.

Base operations

```
base      (n1 --)
          Changes base to n1 so long as it is between 2 and 35.

decimal   (--)
          Changes base to decimal.

binarary  (--)
          Changes base to binary.

octal     (--)
          Changes base to octal.

hex       (--)
          Changes base to hex.
```

4.1.10 Pop up operations

Pop up operations

```
lineChart    (s --)
             Displays a line chart representing the stack.

barChart     (--)
             Displays a bar chart representing the stack.

pieChart     (--)
             Displays a pie chart representing the stack in with
             slices in proportion to its values weight compared to
             the stacks sum.

websearch    ( -- )
             Presents an inbuilt web view with duck duck go as home
             page.

applemap     ( -- )
             Presents apple maps view.
```

4.1.11 Constants and variables

Constants and variables

```
constant (n  --)
         Takes a value from the stack and a name and creates a
         constant of that value called by the name provided.

variable (--)
         Creates a variable whose name will be the first  string of
         characters following variable which is intially zero.

!        (n address --)
         Stores a number to the address provided.

@        (address -- n)
         Gives the value from address

?        (address --)
         Prints the value from address to the terminal.

create   (--)
```

4.1.12 Terminal operators

terminal operators

```
cr          (--)
            Newline added to terminal.

space       (--)
            Adds a  single space to terminal.

spaces      (n --)
            Takes a value from the stack and adds that number of
            spaces to the terminal.

emit        (n --)
            Takes a value from the stack and emits its ascii code.

(           (--)
            Starts comment everything between braces will be
            ignored.

)           (--)
            Ends comment.

.s          (--)
            Show stack on terminal.

.rs         (--)
            Shows return stack.

clear       (--)
            Clears stack and return stack

.           (n -- )
            Takes the last value from the stack and "pops" it
            to terminal.

bye         (--)
            Clears everything, resets to default settings.

."          (--)
            Start print, everything between quotes is printed.

"           (--)
            Ends print.

see         (--)
            Checks dictionary for word and outputs definition.
            -- not implemented yet
```

```
sessionInput (--)
             Displays session input.

dictionary   (--)
             Shows the users defined words, constants and variables.
```

4.1.13 other words

other words

```
date&time (-- second minute hour day month year)
          Adds the devices current time to the stack.

pay       (--)
          Presents pay sheet, where you can send bitcoin.

.fs       (-- n)
          Filename followed by .fs extension will load that files
          contents to Henceforth.
```

4.1.14 Networking operations

Networking operations

```
wocBitcoinPrice       ( -- n )
                      Fetches the bitcoin price in USD $ and adds it
                      to the stack in pennies.

wocCirculatingSupply  (--)
                      -- fix

wocBlockchainInfo     (--)
                      Fetches latest general block info from
                      whats on chain and prints to terminal.

wocAddressInfo        (--)

wocGetBalance         (--)

wocGetHistory         (--)

wocGetBlockByHeight   (n--)
                      Takes a value from the stack and searches
                      Whats on chain for info on that block.

gorillaPoolFeeQuote   (--)
                      Fetches the miner gorrilaPool's fee quotes via
                      MAPI.

taalFeeQuote          (--)
                      Fetches the miner Taal fee quote via MAPI.
```

4.1.15 OPCodes

Push data OP Codes

```
op_0          (--0)
              Zero is pushed on to the stack.

op_1          (--1)
              One is pushed on to the stack.

op_2          (--2)
              Two is pushed on to the stack.

op_3          (--3)
              Three is pushed on to the stack.
```

```
op_4            (--4)
                Four is pushed on to the stack.

op_5            (--5)
                Five is pushed on to the stack.

op_6            (--6)
                Six is pushed on to the stack.

op_7            (--7)
                Seven is pushed on to the stack.

op_8            (--8)
                Eight is pushed on to the stack.

op_9            (--9)
                Nine is pushed on to the stack.

op_10           (--10)
                Ten is pushed on to the stack.

op_11           (--11)
                Eleven is pushed on to the stack.

op_12           (--12)
                Twelve is pushed on to the stack.

op_13           (--13)
                Thirteen is pushed on to the stack.

op_14           (--14)
                Fourteen is pushed on to the stack.

op_15           (--15)
                Fifteen is pushed on to the stack.

op_16           (--16)
                Sixteen is pushed on to the stack.

op_false        (--f) **zero is the flase flag
                Zero is pushed on to the stack. Equal to op_0

op_pushdata1    (--)
                The byte after will contain how many bytes are to be
                pushed on to the stack.

op_pushdata2    (--)
                The next two bytes will contain how many bytes will
                be pushed on to the stack.
```

```
op_pushdata4  (--)
              The next four bytes will contain how many bytes will
              be pushed on to the stack.

op_1negate    (-- -1)
              Minus one is pushed on to the stack.
```

Control flow OP Codes

op_nop	(--) Does nothing.
op_ver	(--) **Disabled Adds the protocol version that the transaction will be evaluated under on to the stack.
op_if	(f--) Executes code depending on the expressions flag.
op_notif	(n--) Executes code depending on the expressions flag.
op_verif	(n--) **Disabled Executes code if the value removed from the stack equals the protocol version the transaction will be evaluated by.
op_vernotif	(--) **Disabled Executes code if the value removed from the stack differs to the protocol version the transaction will be evaluated by.
op_else	(--) Executes if the preceding OP_IF, OP_NOTIF, or OP_ELSE were not excuted.
op_endif	(--) Ends the if else statement.
op_verify	(f--) Marks the transaction as INVALID if the flag is false.
op_return	(--) Ends script. Often used to add data to the blockchain.

Stack operator OP Codes

op_toaltstack	(n--) Takes the top value from the stack and pushes it to the alt stack.
op_fromaltstack	(--n) Takes the top value from the alt stack and pushes it to the stack.

op_2drop	(n n1--) Discards the top 2 elements from the stack.
op_2dup	(n n1 -- n n1 n n1) Duplicates the top two values from the stack adding back to the stack both original and copy.
op_3dup	(n n1 n2 -- n n1 n2 n n1 n2) Duplicates the top three values from the stack adding back to the stack both original and copy.
op_2over	(n n1 n2 n3 -- n n1 n2 n3 n n1) Copies the fourth and third items in the stack over the first and second.
op_2rot	(n n1 n2 n3 n4 n5 -- n2 n3 n4 n5 n n1) Copies the fith and sixth items in the stack over the first and second.
op_2swap	(n n1 n2 n3-- n2 n3 n n1) Takes four values from the stack and swaps their order as pairs.
op_ifdup	(n--) Duplicates the top value from the stack if it is NOT zero
op_depth	(--n) Adds the numbers of items on the stack to the stack.
op_drop	(n--) Discards the top item from the stack.
op_dup	(n--n n) Duplicates the top item from the stack.
op_nip	(n n1--n1) Discards the second item in the stack.
op_over	(n n1--n n1 n) Copies the second item in the stack over the top.
op_pick	(n -- n n) Copies the item n back in the stack.
op_roll	(--) The item n back in the stack is moved to the top.

op_rot	(n n1 n2-- n1 n2 n) The top three items on the stack are rotated left, with the third item going to the top of the stack.
op_swap	(n n1--n1 n) The top two items on the stacks order is reversed.
op_tuck	(n n1--n1 n n1) The top item on the stack is copied in from of the second item.

Splice operator OP Codes

op_cat	(--) Concatenates the two stings provided.
op_split	(n n1--) Splits byte sequence n at n1.
op_num2bin	(n n1--) Converts value n into a byte sequence of length n1.
op_bin2num	(n--) Converts byte sequence n into a numberical value.

bitwise operator OP Codes

op_invert	(n--n) Flips all the bits in the input.
op_and	(n1 n2--AND) Logical AND performed and the result added to the stack.
op_or	(n1 n2--OR) Logial OR performed and the result added to the stack.
op_xor	(n1 n2--XOR) Logical XOR performed and the result added to the stack.
op_equal	(n n1--f) Returns a flag depending on if n1 and n2 are equal. 1 if true 0 if false.
op_equalverify	(n1 n2--f) OP_Equal then OP_Verify
op_reserved1	(--) Renders transaction invalid unless in unexecuted OP_IF statement
op_reserved2	(--) Renders transaction invalid unless in unexecuted OP_IF statement

Arithmetic operator OP Codes

op_1add	(n--n+1) 1 is added to the top value on the stack.
op_1sub	(n--n-1) 1 is subtracted from the top value on the stack.
op_2mul	(n--n*2) The top value on the stack is doubled.
op_2div	(n--n/2) The top value on the stack is halfed.

op_negate	(n-- -n) The top values sign is flipped.
op_abs	(n--|n|) The absolute value of the top item on the stack is returned.
op_not	(n--n) flips the input if 1 or 0 else will return 0.
op_0notequal	(n--f) Returns true if the value is not zero else false.
op_add	(n n1--n3) Top two items from the stack are summed and the value returned.
op_sub	(n n1--n3) the top value from the stack is subtracted from the second value in the stack.
op_mul	(n n1--n3) The top value in the stack is multiplied by the second value in the stack.
op_div	(n n1--n3) The second value in the stack is divided by the top value.
op_mod	(n n1--n3) The second value in the stack is divided by the top value and the remainder returned.
op_lshift	(n n1--n3) The second value in the stack is logical left shifted by the top value.
op_rshift	(n n1--n3) The second value in the stack is logical right shifted by the top value.
op_boolor	(n n1--f) True flag returned if either n or n1 is 0 else false.
op_numequal	(n n1--f) True if n is equivlent to n1 else false.

op_numequalverify	(n n1--f) OP_NUMEQUAL then OP_VERIFY
op_numnotequal	(n n1--f) True if n is NOT equal to n1 else false.
op_lessthan	(n n1--f) True if n is less than n1 else false.
op_greaterthan	(n n1--f) True if n is greater than n1 else false.
op_lessthanorequal	(n n1--f) True if n is less than or equal to n1 else false.
op_greaterthanorequal	(n n1--f) True if n is greater than or equal to n1 else false.
op_min	(n n1--n) The top two values from the stack are taken and the smaller returned.
op_max	(n n1--m) The top two values from the stack are taken and the larger returned.
op_within	(n n1 n2--f) True is returned if n is inbetween n1 and n2

crypto operator OP Codes

op_ripmd160	(--)
op_sha1	(--)
op_sha256	(--)
op_hash160	(--)
op_hash256	(--)
op_codesparator	(--)
op_checksig	(--)
op_checksigverify	(--)
op_checkmultisig	(--)
op_checkmultisigverify	(--)

Lock time operator OP Codes

op_nop2	(--)
op_nop3	(--)
op_pubkeyhash	(--)
op_pubkey	(--)
op_invalidopcode	(--)

reserved OP Codes

op_nop1	(--)
op_nop4	(--)
op_nop5	(--)
op_nop6	(--)
op_nop7	(--)

```
    op_nop8   (--)

    op_nop9   (--)

    op_nop10  (--)
```