

Global

Function	Stack	Notes
1+	n-n	Increment TOS by 1 ¹
1-	n-n	Decrement TOS by 1 ¹
swap	xy-yx	Exchange positions of TOS and NOS ¹
drop	n-	Drop TOS from the stack ¹
and	xy-n	Bitwise AND ¹
or	xy-n	Bitwise OR ¹
xor	xy-n	Bitwise XOR ¹
@	a-n	Fetch a value from a memory location ¹
!	na-	Store a value to a memory location ¹
+	xy-n	Add two values (x+y) ¹
-	xy-n	Subtract two values (x-y) ¹
*	xy-n	Multiply two values (x*y) ¹
/mod	xy-rq	Divide and Remainder ¹ This performs symmetric division
<<	xy-n	Shift bits left (x<<y) ^{1 2}
>>	xy-n	Shift bits right (x>>y) ^{1 2}
tib	-a	Returns address of text input buffer
dup	n-nn	Duplicate TOS ¹
in	p-n	Read a value from an I/O port ¹
out	np-	Write a value to an I/O port ¹
accept	c-	Read a string, ending with the specified character. The string is returned in tib
here	-a	Next free address in heap
,	n-	Place TOS here and increment heap by 1
create	"-	Parse for a name and call header
:	"-	Calls create , changes class to .word , and turns compiler on.
header	\$-	Given a name, create a new header with a class of .data
cr	-	Display a newline character
putc	c-	Display a character
remapKeys	c-c	Remap one ASCII value to another
<puts>	\$-	Helper; default way to display strings
over	xy-xyx	Place a copy of NOS over TOS
]]	-	Turn compiler on
not	x-y	Same as -1 xor; invert TOS and subtract 1.
on	a-	Set a variable to -1 (true)

off	a-	Set a variable to 0 (false)
/	xy-q	Divide two numbers (x/y)
mod	xy-r	Modulus of two numbers (x%y)
negate	x-y	Invert sign of TOS
do	a-	Call a function by address
numbers	-a	Function returning address of string containing all valid numeric characters
wait	-	Wait for an I/O event
'	"-a	Interpret time: return the address ("xt") of a name
@+	a-ac	Fetch a value from an address, return the next address and the value
!+	ca-a	Store a value to an address, return next address
keepString	a-a	Move the string to a permanent location
getLength	a-n	Return the length of a string
withLength	a-an	Same as dup getLength
withClass	ac-	Execute a function via the specified class handler
.word	a-	Class for normal functions
.macro	a-	Class for immediate functions
.data	a-	Class for data (variables, literals, etc)
d->class	a-a	Given a dictionary header, return the address of the class handler. Use @ to get the class handler.
d->xt	a-a	Given a dictionary header, return the address of the function start ("xt"). Use @ to get the actual xt.
d->name	a-a	Given a dictionary header, return the address of the name. This is the actual start of the name.
boot	-	Called when the image first loads; use for custom startup routines
toNumber	\$-n	Convert a string to a number
isNumber?	\$-f	See if a string is a valid number in the current base
ok	-	Displays the "ok" prompt
listen	-	Top level interpreter. Reads and process input.
getc	-c	Read a keypress and return the ASCII value on the stack
find	\$-af	Search for a name in the dictionary. Returns a dictionary header and a flag
notFound	-	Called when a name is not found. Calls <notFound> and displays an error message if necessary
<notFound>	-f	Called by notFound ; hook for custom error handling. Used by the prefix system. Returns a flag of 0 if the error is cleared, or -1 if not
puts	\$-	Display a string ³
compare	\$\$-f	Compare two strings for equality

redraw	-	Update the display. Can be disabled temporarily by update
if	fqq-	Execute first quote if flag is true, second if false
ifTrue	fq-	Execute quote if flag is true
ifFalse	fq-	Execute quote if flag is false
dip	nq-n	Call a quote while temporarily hiding the top item on the stack
sip	nq-n	Call a quote with an item on the stack, restoring that item after the quote returns
=	xy-f	Compare two values for equality
<>	xy-f	Compare two values for inequality
<	xy-f	Compare two values for less than
>	xy-f	Compare two values for greater than
<=	xy-f	Compare for less than or equal to
>=	xy-f	Compare for greater than or equal to
;	-	Compile an exit into a function and stop the compiler
::	-	Compile an exit into a function, but do not stop compilation
repeat	-	Start an unconditional loop
again	-	Jump to the code following the most recent repeat
0;	n-n or n-	If TOS is not zero, do nothing If TOS is zero, drop TOS and exit the function
push	n-	Push a value to the address stack
pop	-n	Pop a value off the address stack
("-	Parse for) and ignore everything it reads
[-	Start a quote (code block)
]	-a	End a quote (code block)
[[-	Turn compiler off
last	-a	Variable; pointer to most recent dictionary header
compiler	-a	Variable; holds compiler state
fb	-a	Variable; Is canvas present? ⁴
fw	-a	Variable; Framebuffer width ⁴
fh	-a	Variable; Framebuffer height ⁴
memory	-a	Variable; Holds amount of memory provided by the VM ⁴
cw	-a	Variable; Console width ⁴
ch	-a	Variable; Console height ⁴
heap	-a	Variable; Pointer to current free location in heap
which	-a	Variable; Holds pointer to most recently looked up header
remapping	-a	Variable; indicates whether CR, LF, and TAB should be treated as whitespace

eatLeading?	-a	Variable; indicates whether accept should ignore leading delimiters
base	-a	Variable; holds current base for numeric conversion and display
update	-a	Variable; flag indicating whether or not redraw should update the display
version	-\$	String holding version information
build	-\$	String holding a build identifier
tabAsWhitespace	-a	Variable; treat tab as whitespace?
nip	xy-y	Drop the NOS from the stack
rot	xyz-yzx	Rotate the top three values on the stack
tuck	xy-yxy	Put a copy of TOS under NOS
+!	na-	Add value to value at address
-!	na-	Subtract value from value at address
++	a-	Increment variable by 1
--	a-	Decrement variable by 1
{{	-	Start a namespace (private portion)
---reveal---	-	Switch to public portion of a namespace
}}	-	Close a namespace, sealing off private symbols
:devector	a-	Restore a function to its original state
:is	aa-	Alter a function to point to a new function
devector	"-	Same as :devector , but parses for name of function
is	a"-	Same as :is , but parses for name of function
default:	"-	Compile call to default definition of a function, ignoring any revectoring
d'	"-a	Parse for a name and return the dictionary header corresponding to it
xt->d	a-d	Given an address, return the corresponding dictionary header or 0 if not found
:hide	a-	Remove a name from a dictionary. Specify the address of a function. Used by hide
hide	"-	Remove a name from the dictionary
reclass	a-	Change class of most recent function to specified class
reclass:	a"-	Same as reclass , but parse for function to change class of
__&	a-a	Prefix; returns address of a variable or function
__@	a-n	Prefix; execute function or data element and fetch from address returned
__!	na-	Prefix; execute function or data element and store value to address returned
__+	na-	Prefix; execute function or data element and add value to value at address returned

__-	na-	Prefix; execute function or data element and subtract value from value at address returned
__2	a-	Prefix; execute function twice
.primitive	a-	Class for functions corresponding to VM opcodes; used for simple optimizations
.compiler	a-	Class for functions that can only be used inside a definition
immediate	-	Set the most recent function to .macro class
compile-only	"-	Set the most recent function to .compiler class
`	"-	Either execute a function, or compile the xt and a call to the corresponding class handler. This will also work with numbers
jump:	"-	Compile a jump to another function
[]	-	Empty quote
while	q-	Execute quote until quote returns a flag of 0
curry	nq-q	5 [.] = [5 [.] do]
take	qq-q	5 [.] = [[.] do 5]
bi	xqq-	Apply each quote to a copy of x
bi*	xyqq-	Apply q1 to x and q2 to y
bi@	xyq-	Apply q to x and y
tri	xqqq-	Apply each quote to a copy of x
tri*	xyzqqq-	Apply q1 to x, q2 to y, and q3 to z
tri@	xyzq-	Apply q to x, y, and z
cons	ab-q	Create a quote returning two data elements
preserve	aq-	Given a variable (a) and a quote (q), preserve the contents of (a) while executing the quote, and restore the original contents of (a) after execution completes. (a) is removed from the stack before (q) is executed.
when	nqq-n	Execute q1, with a copy of n on the stack. If q1 returns a true flag, run q2 and exit the caller. If not, discard q2 and return to the caller. q2 is permitted to discard n, which will alter the stack effect.
whend	nqq-?	Execute q1, with a copy of n on the stack. If q1 returns a true flag, drop n, run q2 and exit the caller. If not, discard q2 and return to the caller.
times	nq-	Run quote (n) times
iterd	nq-	Run quote (n) times and push counter to stack each time. Counts down.
iter	nq-	Run quote (n) times and push counter to stack each time. Counts up.
<each@>	...t-	Hook into each@ for adding additional types

each@	...t-	Supercombinator for applying quote to each item in various data structures. Also provide on the stack:										
		<table border="1"> <thead> <tr> <th>Type</th> <th>Stack</th> </tr> </thead> <tbody> <tr> <td>ARRAY</td> <td>aq-</td> </tr> <tr> <td>BUFFER</td> <td>anq-</td> </tr> <tr> <td>STRING</td> <td>\$q-</td> </tr> <tr> <td>LIST</td> <td>lq-</td> </tr> </tbody> </table>	Type	Stack	ARRAY	aq-	BUFFER	anq-	STRING	\$q-	LIST	lq-
		Type	Stack									
		ARRAY	aq-									
		BUFFER	anq-									
STRING	\$q-											
LIST	lq-											
For LIST, / should be a variable pointing to the actual list.												
The quote is given the address of the current element each time it is invoked.												
copy	aan-	Copy n values from source (a1) to dest (a2)										
fill	ann-	Fill (n2) memory locations starting at (a) with value (n1)										
ahead	-a	Used in conditionals; compiles a branch to be patched in later										
if;	f-	Exit function if TOS is a non-zero flag										
within	xlu-f	Is (x) within lower (l) and upper (u) bounds?										
variable:	n"-	Create a new variable with an initial value										
variable	"-	Create a new variable with an initial value of 0										
constant	n"-	Create a numeric constant										
string	\$"-	Create a string constant										
allot	n-	Allocate space in the heap										
elements	n"-	Create a series of variables										
decimal	-	Switch base to 10										
hex	-	Switch base to 16										
octal	-	Switch base to 8										
binary	-	Switch base to 2										
toString	n-\$	Convert a number into a string										
clear	-	Clear the display										
space	-	Display a space character (ASCII 32)										
putn	n-	Display a number										
.parse	a-	Class for parsing prefixes										
parsing	-	Set most recent function to .parse class										
__\$	\$-n	Prefix; treat number as hexadecimal (base 16)										
__#	\$-n	Prefix; treat number as decimal (base 10)										
__%	\$-n	Prefix; treat number as binary (base 2)										
__'	\$-n	Return character following '										
dicts	-a	Array; used by chained vocabularies and search order code										
%%	-	Close a vocabulary. Use with caution										
<%>	a-	Open a vocabulary. Use with caution										
.chain	a-	Class for vocabularies										

chain:	"-	Create a new vocabulary
;chain	-	End a vocabulary
:with	a-	Add a vocabulary to the search order (by pointer)
with	"-	Add a vocabulary to the search order (parses for name)
without	-	Remove a vocabulary from the search order
global	-	Remove all vocabularies from the search order, leaving just the global dictionary
findInChain	\$a-df	Open a chain (using :with) and search for a name. Closes the chain when done.
with	"-	Open a series of vocabularies, ending when is encountered
rename:	a"-	Rename a function
STRING-LENGTH	-n	Return the max length for a string
STRING-BUFERS	-n	Return number of temporary string buffers
tempString	a-a	Move a string to a temporary buffer
__"	"-\$	Prefix; parse and return a string
"	"-\$	Parse and return a string
formatted	-a	Variable; toggles whether puts uses escape sequences or not
depth	-n	Return number of items on stack
reset	...-	Remove all items from stack
.s	-	Display all items on stack
words	-	List all names in dictionary
save	-	Save the image
bye	-	Exit Retro
getToken	"-\$	Read a string, stopping at first whitespace
getNumber	"-n	Read a number from the input stream
:include	\$-	Include a file
include	"-	Same as :include , but parse for file name
time	-n	Return the current unix time
delay	n-	Delay for (approximately) n seconds
getEnv	a\$-	Get a copy of environment variable \$ in buffer
later	-	Defer execution of caller until a later time
__^	"-	Allow direct access to functions in a chain
needs	"-	Load a vocabulary from the <i>library</i> if it is not already loaded ⁵
doc{	"-	Parse tokens up to }doc and ignore. This is intended as a means of embedding docs into libraries.
variables	"-	Create a series of variables
pow	bp-n	Raise (b) to power (p)
abs	n-n	Absolute value of number (n)
min	ab-c	Minimum of (a) or (b)

max	ab-c	Maximum of (a) or (b)
random	-x	Return a random number

buffer'

Function	Stack	Notes
start	-a	Get starting address of buffer
end	-a	Address at end of buffer
add	c-	Add value to end of buffer
get	-c	Read and remove value from buffer
empty	-	Remove everything from the buffer
size	-n	Number of values in buffer
set	a-	Set buffer to memory address and empty it

strings'

Function	Stack	Notes
search	\$\$-f	Search for a string (2) within a string (1); return string starting with substring
findChar	\$c-a	Search for a character within a string; return string starting at the character
chop	\$\$-	Return a new string, with the last byte removed
getSubset	\$nn-\$	Return a subset of (\$) starting at (n1) with length of (n2)
trimLeft	\$\$-	Trim whitespace from left side of string
trimRight	\$\$-	Trim whitespace from right side of string
append	\$\$-\$	Append second string to first
appendChar	\$c-\$	Append character to a string
prepend	\$\$-\$	Append first string to second
toLower	\$\$-	Convert a string to all lower case
toUpper	\$\$-	Convert a string to all upper case
reverse	\$\$-	Reverse the characters in a string; returns a new string
split	\$n-\$\$	Split a string into two parts
splitAtChar	\$c-\$\$	Search for a character and return two strings (up to and including (c), and after (\$2))
splitAtChar:	\$"-\$\$	Parse for a character and call splitAtChar

files'

Function	Stack	Notes
:R	-n	Mode for reading a file. Does not create file

:W	-n	Mode for writing a file
:A	-n	Mode for appending to a file
:M	-n	Mode for modifying a file. Does not create file.
open	\$m-h	Open a file. Will return a handle. Valid handles will be non-zero. A zero handle indicates failure to open a file.
read	h-c	Read a byte from a file. This returns the byte.
write	ch-f	Write a byte to a file. This returns a flag indicating the number of bytes written. (Should always equal '1')
close	h-f	Close an open file. Returns a flag of zero if unable to close, or non-zero if successful.
pos	h-n	Get current position in a file
seek	nh-f	Seek a position in a file
size	h-n	Return size of open file
delete	\$-f	Delete a file. Returns a handle. Non-zero if successful, zero if failed.
slurp	a\$n	Read a file into a buffer
spew	an\$n	Write (n) bytes from address (a) into a file
readLine	h-\$	Read a line from a file
writeLine	\$h-	Write a string to a file

types'

Function	Stack	Notes
ARRAY	-n	Type constant for arrays
BUFFER	-n	Type constant for buffers
STRING	-n	Type constant for strings
LIST	-n	Type constant for linked lists

Footnotes

Sequence	Stack	Used For
%s	\$-	Display a string
%d	n-	Display a number (decimal)
%x	n-	Display a number (hexadecimal)
%o	n-	Display a number (octal)
%c	c-	Display a character
\n	-	Newline
\'	-	Display a double quote (")
\[-	Start an ANSI escape sequence

-
- 1(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18) This corresponds to an Ngaro instruction.
- 2(1, 2) Shifting by a negative amount is undefined behavior. Don't do it.
- 3 **puts** supports escape sequences which alter the stack usage.
- 4(1, 2, 3, 4, 5, 6) These are updated each time the image is loaded.
- 5 This parses for a vocabulary name, which should end in a single apostrophe. The apostrophe will be cut, and the a suffix of *.rx* added. The system will attempt to load the file from the *library* subdirectory in the current working directory.