Ngaro VM Implementation Status

Language	File(s)	A	B	C	D	E	F	G	H
Assembly	retro.s	x	x	x	x	x	x	x	x
C	retro.c	x	x	x	x	x	x	x	x
C	libretro.c	x	x	x	x	x	x	x	x
C#	retro.cs	x	x		x	X	x	x	
F#	retro.fsx	x	x	x	x	x	x	x	x
Forth	retro.fs	x	x	x		x	x	x	x
Go	go/	x	x	x	x	x	x	x	x
Lisp	lisp/	x	x	x	x	x	x	x	x
Java	retro.java	x				x	x	x	
Lua	retro.lua	x				x	x	x	
Perl	retro.pl	x				x	x	x	
PHP	retro.php	x	x	x		x	x	x	x
Python	retro.py	x	x	x	x	x	x	x	x
Retro	retro.rx	x	x	x	x	x	x	x	x
Ruby	retro.rb	x	x	x	x	x	x	x	x
Scheme	retro.scm						x	x	
JavaScript	html5/*	x				x	x	x	

A. save image

B. **include**, **:include**, **needs**

C. file i/o

D. **getEnv**

E. **time**

F. passes core test suite

G. passes vocab test suite

H. passes file io test suite

Notes on Specific Implementations

Assembly

This is for 32-bit x86 Linux systems.

If you have problems with the input (e.g., under PuTTY), try:

```
cat - | ./retro retroImage
```

C

The C implementation can be built to support use of 16-bit, 32-bit, or 64-bit images. The default is to build for standard 32-bit images.

To build for 16-bit, define **RX16**, and for 64-bit, define **RX64**. To actually get a 16-bit or 64-bit image, do:

```
make tools
./convert
```

Forth

The Forth VM is more of a virtualizer than an actual virtual machine. It maps the image to the host memory, and uses the standard Forth data stack and some native functions. As such it needs to be used with an image that matches your Forth's cell size.

There is no support for **getEnv** due to lack of a standard function to access this in ANS Forth.

You can build 16-bit and 64-bit images using the *convert* tool.

Lua

The Lua implementation requires Lua JIT *or* Lua 5.2. Earlier releases of Lua lack the bitwise operators. Note that a single line needs to be altered in the source to enable use under Lua 5.2.

JavaScript

This implementation does not support **include**, **needs**, **getEnv**, or the **files'** vocabulary. The scripts used to build the retroImage.js now hide these functions.

Embedded Implementations

Retro can be used on some development boards. There may be limitations to the overall functionality, or special extensions to use features of the target platforms. See the individual implementations for notes.

CPU	Board
ARM	mbed.org
AVR	arduino

Experimental Implementations

These are all either non-standard in some way or remain mostly untested in real-world applications.

Language	File(s)	Description
C	retroht.c	Faster implementation using a hybrid threading model.
C	retro-dynamic.c	Provides command line flags for altering stack depth and memory size
C	sdl/	SDL 1.3 based Ngaro w/support for canvas
Python	retro-skinless.py	Python implementation w/o file I/O for use in testing the ShedSkin compiler