```
------------------------------------------------------------------------
------------
  Test suite for Mecrisp-Ice
------------------------------------------------------------------------
------------

Idea:

The Forth core is executed in an emulator, which records the
execution, read and write accesses separately for each memory
location.

In this environment, the (extended) Hayes-Forth tester is executed,
which checks the definitions for their standard-compliant function
and all operators for correct results, whereby trivial cases are
also recorded and checked.
This Hayes test suite is logically structured so that, as far as
possible, the tests only use primitives that have already been
tested correctly.

https://github.com/gerryjackson/forth2012-test-suite/

I have adapted and extended this test suite to the special features of
Mecrisp-Ice.

After the tests have been run, the emulator checks the memory image
for locations that have neither been read nor executed and can
assign these to the individual definitions using a rudimentary
interpretation of the dictionary structure. This allows the test
coverage to be checked and completed.

There are some routines in the Forth core that can only be executed
meaningfully when used manually (e.g. pressing the delete key when
typing commands), only in the event of an error (e.g. aborting the
program) or only in real hardware (IO registers, interrupts) and are
therefore not covered by the automatic tests. However, these cases
are manageable, can be tried out in real hardware and were found to
be good.

The entire test suite can also be simulated on the logic of the
"real" processor in the Verilator. This allows the test results in
the emulator to be compared with the actual implementation of the
processor in logic and errors in the processor source code itself
can be ruled out, at least insofar as they change the functionality
covered by the test. In the Verilator, however, juggling with the
memory image itself is more complicated, so the test coverage check
is carried out in the emulator.

------------------------------------------------------------------------
------------
  Use
------------------------------------------------------------------------
------------
```

First of all, compile the core with all additions and find out the desired dictionary start address:

```
./emulate

insight

[ESC]
```

The address of "new" is the limit of the basic configuration and therefore a good starting point. In this case 0x1CAC.

This address must be re-entered in the "testing" script after each change in Forth.

Then start ./testing.

After a while, this issue appears:

```
------------------------------------------------------------------------
------------
  Untested routines:
------------------------------------------------------------------------
----------
1C9E Code:  1CA4  - 1CAA Unknown:  0004  Name: see
1C54 Code:  1C5C  - 1C9C Unknown:  0021  Name: seec
1B7C Code:  1B8A  - 1C52 Unknown:  0065  Name: disasm-step
1B58 Code:  1B64  - 1B7A Unknown:  000C  Name: memstamp
1AB0 Code:  1AB8  - 1B56 Unknown:  0050  Name: alu.
19F2 Code:  19FA  - 1AAE Unknown:  005B  Name: name.
19DE Code:  19EC  - 19F0 Unknown:  0003  Name: disasm-cont
19CC Code:  19D8  - 19DC Unknown:  0003  Name: disasm-$
18A4 Code:  18AA  - 1912 Unknown:  001A  Name: .s
183A Code:  1842  - 18A2 Unknown:  0031  Name: dump
1744 Code:  174E  - 1754 Unknown:  0002  Name: divisor
1734 Code:  173C  - 1742 Unknown:  0002  Name: shift
1720 Code:  172C  - 1732 Unknown:  0002  Name: dividend
13E6 Code:  13F4  - 1408 Unknown:  000B  Name: cornerstone
11DC Code:  11E2  - 11E6 Unknown:  0002  Name: hld
11D2 Code:  11D8  - 11DA Unknown:  0002  Name: BUF
1146 Code:  114E  - 11D0 Unknown:  001E  Name: BUF0
0F54 Code:  0F5C  - 0F66 Unknown:  0006  Name: xor!
0F3C Code:  0F44  - 0F52 Unknown:  0008  Name: bic!
0F28 Code:  0F30  - 0F3A Unknown:  0006  Name: until!
0F18 Code:  0F20  - 0F26 Unknown:  0004  Name: unites?
0F0E Code:  0F16  - 0F16 Unknown:  0001  Name: dint
0F04 Code:  0F0C  - 0F0C Unknown:  0001  Name: unites
0EC8 Code:  0ED0  - 0F02 Unknown:  0001  Name: quit
0E1E Code:  0E28  - 0E82 Unknown:  0007  Name: accept
0C24 Code:  0C2E  - 0E1C Unknown:  0015  Name: number
0B78 Code:  0B80  - 0B86 Unknown:  0001  Name: abort
057E Code:  0584  - 0586 Unknown:  0002  Name: tib
0572 Code:  057A  - 057C Unknown:  0002  Name: init
```

```
0566 Code: 056E - 0570 Unknown:  0002  Name: forth
048C Code: 0496 - 0542 Unknown:  0011  Name: /string
026E Code: 0274 - 0274 Unknown:  0001  Name: io@
0264 Code: 026A - 026C Unknown:  0002  Name: io!
------------------------------------------------------------
------------
  Errors found:
------------------------------------------------------------
------------
           = 0= IF S" INCORRECT RESULT: " ERROR LEAVE THEN ok.
      S" WRONG NUMBER OF RESULTS: " ERROR ok.
```

At the very end, all lines containing INCORRECT or WRONG, the two possible error messages of the test suite, are listed. If only these two lines in which the error messages are defined are recognized, everything is fine. Details can be viewed in log.txt.

The "Unknown" column indicates how many opcodes in the respective definition were not executed or read during the test run.
These definitions must therefore be considered individually:

The first block are debug tools that only generate output for the user and as such cannot be tested automatically, but also do not perform a critical function in the Forth system:

```
1C9E Code: 1CA4 - 1CAA Unknown: 0004  Name: see
1C54 Code: 1C5C - 1C9C Unknown: 0021  Name: seec
1B7C Code: 1B8A - 1C52 Unknown: 0065  Name: disasm-step
1B58 Code: 1B64 - 1B7A Unknown: 000C  Name: memstamp
1AB0 Code: 1AB8 - 1B56 Unknown: 0050  Name: alu.
19F2 Code: 19FA - 1AAE Unknown: 005B  Name: name.
19DE Code: 19EC - 19F0 Unknown: 0003  Name: disasm-cont
19CC Code: 19D8 - 19DC Unknown: 0003  Name: disasm-$
18A4 Code: 18AA - 1912 Unknown: 001A  Name: .s
183A Code: 1842 - 18A2 Unknown: 0031  Name: dump
```

These three definitions are variables whose addresses have been compiled in in an optimized way at the points where they are used, so that the variable names are no longer needed during the tests themselves. However, they have proven their function by the fact that the routines UD/MOD and F/ based on them work perfectly.

```
1744 Code: 174E - 1754 Unknown: 0002 Name: divisor
1734 Code: 173C - 1742 Unknown: 0002 Name: shift
1720 Code: 172C - 1732 Unknown: 0002 Name: dividend
```

This is a tool for setting a return point and deleting subsequent parts from the dictionary.

```
13E6 Code: 13F4 - 1408 Unknown: 000B Name: cornerstone
```

Five ways to access IO ports - however, these are not connected to any hardware in the emulator, so they are not tested automatically. However, they have been tested manually in real hardware.

026E Code: 0274 - 0274 Unknown: 0001 Name: io@
0264 Code: 026A - 026C Unknown: 0002 Name: io!
0F54 Code: 0F5C - 0F66 Unknown: 0006 Ｎａｍｅ：
xor! 0F3C Code: 0F44 - 0F52 Unknown: 0008 Name:
bic! 0F28 Code: 0F30 - 0F3A Unknown: 0006 Name:
bis!

Interrupts. Also only useful to test in real hardware, and functional there.

0F18 Code: 0F20 - 0F26 Unknown: 0004 Name: eint?
0F0E Code: 0F16 - 0F16 Unknown: 0001 Name: dint
0F04 Code: 0F0C - 0F0C Unknown: 0001 Name: eint

Internal variables of the Forth kernel whose addresses have been compiled in at the appropriate places in an optimized manner when the kernel was created and are no longer used afterwards.

057E Code: 0584 - 0586 Unknown: 0002 Name: tib
0572 Code: 057A - 057C Unknown: 0002 Name: init
0566 Code: 056E - 0570 Unknown: 0002 Name: forth
11DC Code: 11E2 - 11E6 Unknown: 0002 Name: hld
11D2 Code: 11D8 - 11DA Unknown: 0002 Name: BUF

This here is a buffer for the strings during the number output, which was not utilized to its limits during the test execution, so that some memory locations remained unused.

1146 Code: 114E - 11D0 Unknown: 001E Name: BUF0

Quit and Abort both terminate the execution of the current program, automatic test not fully possible.

0EC8 Code: 0ED0 - 0F02 Unknown: 0001 Name: quit
0B78 Code: 0B80 - 0B86 Unknown: 0001 Name: abort

Accept is responsible for the entries in the terminal - the opcodes that are not automatically tested are responsible for handling backspace, i.e. for correcting typing errors and for limiting the maximum input length.

0E1E Code: 0E28 - 0E82 Unknown: 0007 Name: accept

At the end of NUMBER and /STRING there are still internal routines of the compiler for the optimizations, which do not have their own name in the dictionary structure and do not become active during the tests. Due to the type of dictionary structure and its location, these routines are nevertheless counted as part of NUMBER, which would otherwise be called

has been tested.

0C24 Code: 0C2E - 0E1C Unknown: 0015 Name: number
048C Code: 0496 - 0542 Unknown: 0011 Name: /string

  Removed from the official test suite and/or deviations from the standard
---

* VARIABLE and 2VARIABLE are initialized in Mecrisp-Ice, so they are redefined as uninitialized in preparation for running the test suite:

```
: variable ( -- ) 0 variable ;
: 2variable ( -- ) 0. 2variable ;
```

* Since variables and constants are optimized internally, they should not be flagged with IMMEDIATE, which would be permitted according to the standard.

* NUMBER from Mecrisp-Ice has a completely different internal structure than
>NUMBER from the standard and allows the mixing of base prefixes in a number as well as the entry of fixed-point numbers with decimal places.

* FIND works very differently from the standard due to the special flags for the optimizations.

* Since Mecrisp-Ice is word-addressed, there is no "C" for appending individual bytes to the dictionary. C" for counted strings is also missing. The corresponding passages in the test suite have therefore been commented out.

* MARKER VALUE TO 2VALUE DEFER and many string routines do not exist in Mecrisp-Ice and would only waste space there.

* 2>R 2R> 2R@ ROLL PICK are defined for the test suite at the beginning, but are rather cumbersome due to the available command set and its functionality and are not recommended on this architecture.

  Comparison of the official and the modified test suite

From the forth2012-test-suite-master/src/ directory: kompare

```
core.fr ../../test-core.txt
compare coreplustest.fth ../../test-core-plus.txt
```

```
compare coreexttest.fth ../../test-core-ext.txt
compare doubletest.fth ../../test-double.txt
compare stringtest.fth ../../test-strings.txt

Without equivalent: test-mecrisp-ice-extras.txt
```