Testsuite für Mecrisp-Ice

Idee:

Der Forth-Kern wird in einem Emulator ausgeführt, welcher für jede Speicherstelle separat die Ausführ-, Lese- und Schreibzugriffe aufzeichnet.

In dieser Umgebung wird der (erweiterte) Hayes-Forth-Tester ausgeführt, womit die Definitionen auf ihre standardkonforme Funktion und sämtliche Operatoren auf korrekte Ergebnisse überprüft werden, wobei auch triviale Fälle erfasst und überprüft werden. Diese Hayes-Testsuite ist logisch aufeinander aufbauend ausgerichtet, so dass die Tests so weit wie irgend möglich nur auf bereits korrekt getestete Primitiven zurückgreifen.

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

Diese Testsuite wurde von mir an die Besonderheiten von Mecrisp-Ice angepasst und erweitert.

Nachdem die Tests gelaufen sind, prüft der Emulator das Speicherabbild auf Stellen, die weder gelesen noch ausgeführt worden sind, und kann diese mit einer rudimentären Interpretation der Dictionarystruktur den einzelnen Definitionen zuordnen. Damit lässt sich die Testabdeckung prüfen und vervollständigen.

Es gibt einige Routinen im Forth-Kern, die nur bei manueller Benutzung (z.B. Betätigung der Löschtaste beim Tippen von Befehlen), nur im Fehlerfalle (z.B. Abbruch des Programms) oder nur in echter Hardware sinnvoll ausgeführt werden können (IO-Register, Interrupts) und somit nicht von den automatischen Tests erfasst werden. Diese Fälle sind jedoch überschaubar, können in echter Hardware ausprobiert werden und wurden für gut befunden.

Die gesamte Testsuite kann auch auf der Logik des "echten" Prozessors simuliert im Verilator ausgeführt werden. Dadurch lassen sich die Testergebnisse im Emulator mit der tatsächlichen Implementierung des Prozessors in Logik vergleichen und es können Fehler im Prozessorquelltext selbst ausgeschlossen werden, zumindest insofern sie dessen vom Test abgedeckte Funktionsweise verändern. Im Verilator ist jedoch das Jonglieren mit dem Speicherabbild selbst komplizierter, so dass die Probe der Testabdeckung im Emulator durchgeführt wird.

Benutzung

Zuerst einmal den Kern mit allen Zusätzen kompilieren und die gewünschte Dictionarystartadresse herausfinden:

```
./emulate

insight

[ESC]
```

Die Adresse von "new" ist die Grenze der Grundausstattung und somit ein guter Anfangspunkt. In diesem Fall 0x1CAC.

Diese Adresse muss nach jeder Änderung im Forth neu in das "testing"-Skript eingetragen werden.

Dann ./testing starten.

Nach einer Weile erscheint diese Ausgabe:

```
-------------------------------------------------------------------------------
  Ungetestete Routinen:
-------------------------------------------------------------------------------
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: bis!
0F18 Code: 0F20 - 0F26 Unknown: 0004  Name: eint?
0F0E Code: 0F16 - 0F16 Unknown: 0001  Name: dint
0F04 Code: 0F0C - 0F0C Unknown: 0001  Name: eint
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!
-------------------------------------------------------------------
-----------
  Gefundene Fehler:
-------------------------------------------------------------------
-----------
           = 0= IF S" INCORRECT RESULT: " ERROR LEAVE THEN  ok.
      S" WRONG NUMBER OF RESULTS: " ERROR  ok.
```

Ganz am Ende werden alle Zeilen aufgelistet, die INCORRECT oder WRONG beinhalten, die beiden möglichen Fehlermeldungen der Testsuite. Werden ausschließlich diese beiden Zeilen erkannt, in denen die Fehlermeldungen definiert werden, ist alles prima. Details sind in log.txt einzusehen.

Die Spalte "Unknown" gibt an, wie viele Opcodes in der jeweiligen Definition während des Testlaufes nicht ausgeführt oder gelesen worden sind.
Diese Definitionen müssen deshalb einzeln näher betrachtet werden:

Der erste Block sind Debug-Werkzeuge, die ausschließlich Ausgaben für den Benutzer generieren und als solche nicht automatisch getestet werden können, allerdings auch keine kritische Funktion im Forth-System übernehmen:

```
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
```

Bei diesen drei Definitionen handelt es sich um Variablen, deren Adressen an den Stellen, wo sie verwendet werden, optimiert einkompiliert wurden, so dass die Variablennamen während der Tests selbst nicht mehr benötigt werden. Sie haben jedoch ihre Funktion dadurch bewiesen, dass die darauf aufbauenden Routinen UD/MOD und F/ einwandfrei funktionieren.

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

Hierbei handelt es sich um ein Werkzeug, um einen Rückkehrpunkt zu setzen und darauf folgende Teile aus dem Dictionary zu löschen.

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

Fünf Möglichkeiten, auf IO-Ports zuzugreifen - diese sind im Emulator jedoch nicht an irgendwelche Hardware angeschlossen, so dass sie nicht automatisch getestet werden. Sie wurden jedoch manuell in echter Hardware getestet.

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. Ebenfalls nur in echter Hardware sinnvoll zu testen, und dort funktionstüchtig.

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

Interne Variablen des Forth-Kernes, deren Adressen beim Erstellen des Kernes an den entsprechenden Stellen optimiert einkompiliert worden sind und danach nicht mehr verwendet werden.

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

Dieses hier ist ein Puffer für die Strings während der Zahlenausgabe, der während der Testausführung nicht bis an seine Grenzen ausgelastet worden ist, so dass darin einige Speicherstellen unbenutzt geblieben sind.

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

Quit und Abort beenden beide die Ausführung des aktuellen Programmes, automatischer Test nicht vollständig möglich.

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

Accept ist für die Eingaben im Terminal zuständig - die nicht automatisch getesteten Opcodes sind für die Behandlung von Backspace zuständig, also zur Korrektur von Tippfehlern und für die Begrenzung beim Erreichen der maximalen Eingabelänge.

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

Am Ende von NUMBER und /STRING befinden sich noch interne Routinen des Compilers für die Optimierungen, die keinen eigenen Namen in der Dictionarystruktur besitzen, und während der Tests nicht aktiv werden. Durch die Art der Dictionarystruktur und ihre Stelle werden diese Routinen trotzdem zu NUMBER gezählt, welches ansonsten

getestet worden ist.

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

  Aus der offiziellen Testsuite entfernt und/oder Abweichungen zum Standard
---

* VARIABLE und 2VARIABLE sind in Mecrisp-Ice initialisiert, deswegen werden sie als Vorbereitung für den Durchlauf der Testsuite als uninitialisiert redefiniert:

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

* Dadurch dass Variablen und Konstanten intern optimiert werden, sollten sie nicht mit IMMEDIATE geflaggt werden, was laut Standard erlaubt wäre.

* NUMBER von Mecrisp-Ice hat einen ganz anderen inneren Aufbau als >NUMBER aus dem Standard und erlaubt die Mischung von Basisprefixen in einer Zahl sowie die Eingabe von Fixkomma-Zahlen mit Nachkommastellen.

* FIND funktioniert auf Grund der speziellen Flags für die Optimierungen ganz anders als im Standard.

* Da Mecrisp-Ice wordadressiert ist, gibt es kein C, zum Anhängen einzelner Bytes ans Dictionary. Ebenso fehlt C" für counted strings. Die entsprechenden Passagen in der Testsuite wurden deshalb auskommentiert.

* MARKER VALUE TO 2VALUE DEFER und viele Stringroutinen gibt es in Mecrisp-Ice nicht und würden dort nur Platz verschwenden.

* 2>R 2R> 2R@ ROLL PICK werden für die Testsuite zu Beginn definiert, sind aber auf Grund des verfügbaren Befehlssatzes und dessen Funktionsweise eher umständlich und auf dieser Architektur nicht empfehlenswert.

  Vergleich der offiziellen und der modifizierten Testsuite

Aus dem Verzeichnis forth2012-test-suite-master/src/ heraus:

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

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

Ohne Entsprechung: test-mecrisp-ice-extras.txt
```