RCSL No: 99 0 00913

Edition: May, 1986

Author: Henning Jakobsen

Title:

RC3502 CPU Testprogram Package
User's Guide

PN: 99200170

RC International

Keywords:

RC3502, Central Processor Testprogram.

Abstract:

This manual contains a reliability test of the RC3502 CPU. It has been divided into three sections, one testing the microinstruction set, and the other testing the working registers and some ALU functions.

(22 printed pages).

Printed by A/S Regnecentralen af 1979, Copenhagen

PN: 99200170

TABLE OF CONTENTS PAGE

1. INTRODUCTION

This manual describes the RC3502 model 2 CPU reliability test. The testprogram is divided into three different sections, which run consecutive.

The first ("a") is the most differentiated. It is testing a large amount of the microinstructions, except the monitor based instructions as SIGNAL and WAIT. The strategy of the test is that no instruction not yet tested will be used for testing others (as far as possible).

The second part ("b") is testing the communication with the debug console, and some selected ALU-functions.

The thirth part ("c") is testing the scheduler, all registersets, process creation, process communication (SIGNAL, WAIT), and removal of processes.

The CPU-test is a testpackage in the RC3502 test system, TOP35, but it is deviating from the other testpackages, because it can be started directly by OPSYS.

This is done to have as few demands as possible to the capabilities of the system, before running this test.

Although keep in mind still that this is a verification of the CPU operationability and not a diagnostic test, and should be regarded as such. Also meaning that though the test contains routines for writing some readable texts, it could well be the case that you would not have any hint of an occured error in the CPU from this test (the principle of go or no-go).

It is written in REAL-TIME PASCAL (RC3502 model 2 implementation).

1.1 Configuration Requirements 1.1

A minimum configuration RC3502, consisting of CPU, at least two memory modules, and a load possibility for the test (e.g. RC8000 connections or TES modules with the testsystem) and a debug console.

1.2 Parameter Values 1.2

The test has no selective parameters. Even the number of runs cannot be selected, meaning that the test will run forever and has no terminating point.

1.3 Load and Start of the Test 1.3

How to load in general, see ref. 1. The CPU-test is loaded as described in ref. 2.

When the TOP35 is loaded, the CPU-test is started by the command:

```
NEW:CPU
```

which causes the test to be started without any further commands. Output is generated after 2 minutes. See chapter 2 for description of the running test.

It is also possible to start the test directly under OPSYS, without using the TOP35 system. The CPU-test is started by the following commands:

```
>OPSYS
RUN CPU
```

Observe that it will have no meaning to start more than one incarnation of the CPU-test.

Chapter 3 describes how to terminate the CPUTEST.

2. TEST SPECIFICATION

2.1 Microinstruction Test

This part of the testprogram is testing the micro instructions of the CPU microprogram. Monitor based instructions such as SIGNAL and WAIT are tested in the scheduler test.

In the following, the instructions tested are described with related mnemonic codes and trace numbers. It is the same codes and numbers which will be printed on the console if an error is detected. The mnemonic codes are referring to the same codes of the LMI-text. The trace numbers are telling, which testloop went wrong. They are in the range from 0 to 72.

2.1.1 Jump

Jump on true condition:
mnemonic code: jmzeq
trace No: 1

Do not jump on false condition:
mnemonic code: jmzeq
trace No: 2

Jump on inverted false condition:
mnemonic code: notinstr
trace No: 3

Do not jump on inverted true condition:
mnemonic code: notinstr
trace No: 4

2.1.2 Not, And, Or, Equal

Is (NOT false) = true:
mnemonic code: notinstr
trace No: 5

Is (true AND true) = true:
mnemonic code: andinstr
trace No: 6

Is (NOT (true AND false)) = true:
mnemonic code: andinstr
trace No: 7

Is (NOT (false AND true)) = true:
mnemonic code: andinstr
trace No: 8

Is (NOT (false AND false)) = true:
mnemonic code: andinstr
trace No: 9

Is ((true OR true) AND (true OR false) AND (false OR true) AND (NOT (false OR false))) = true:
mnemonic code: orinstr
trace No: 10

Is a variable earlier assigned to 1, equal 1 and not 0:
mnemonic code: eq
trace No: 11

Is the not equal instruction working on two different variables:
mnemonic code: ne
trace No: 12

2.1.3 Case 2.1.3

Testing the case jump instruction with switches from 0 to 3:
mnemonic code: jmcht
trace No: from 13 to 16

2.1.4 Not Equal, Greater, Less 2.1.4

These instructions are tested on two variables, which will have all combinations of the following values:
- 32768, -32767, -6, -5, -4, -1, 0, 1, 4, 5, 6, 32766, 32767

not equal:
mnemonic code: ne
trace No: 17

greater than or equal:

mnemonic code: ge
trace No: 18

less than:
mnemonic code: lt
trace No: 19

greater than:
mnemonic code: gt
trace No: 20,21

less than or equal:
mnemonic code: le
trace No: 22

2.1.5 Negation

2.1.5

Testing negation of two different variables (0,1):
mnemonic code: neg
trace No: 23

2.1.6 Index

2.1.6

The index instruction is tested by writing indexed in a byte- layout an checking in an indexed bit-layout bit by bit. The loop is repeated with different length of the area to be indexed. The length variates from 2 bytes to 256 bytes.

mnemonic code: index
trace No: 24

2.1.7 Retrieve and Store Addressing

2.1.7

The CPU can address the memory with different instructions:

retreive, store local (word):
mnemonic code: revlw
trace No: 25

retreive, store intermidiate (word):

mnemonic code: revsw
trace No: 26

retreive, store intermidiate (word):
mnemonic code: reviw
trace No: 27

Retreive, store global (word):
mnemonic code revgw
trace No: 28

2.1.8 Combinations of Retrieve and Store

2.1.8

It is tested that data stored by either word, byte or field instructions can be retrieved by either word, byte or field instructions.

store words, retrieve words:
mnemonic code: stvsw.revsw
trace No: 29

store words, retrieve bytes:
mnemonic code: stvsw.revsb
trace No: 30, 31

store words, retrieve field:
mnemonic code: stvsw.revsf
trace No: 31

store bytes, retrieve words:
mnemonic code: stvsb.revsw
trace No: 32

store bytes, retrieve bytes:
mnemonic code: stvsb.revsb
trace No: 33, 34

store bytes, retrieve field:
mnemonic code: stvsb.revsf
trace No: 34

store field, retrieve words:
mnemonic code: stvsf.revsw
trace No: 35

store field, retrieve bytes:
mnemonic code: stvsf.revsb
trace No: 36, 37

store field, retrieve field:
mnemonic code: stvsf.revsf
trace No: 37

2.1.9 Multiplicate, Divide

It is tested that the result of the divide, division remainder and multiplication instructions is correct when calculated with different factors.

The expressions tested are: (i div 5 = q) AND (i div (-5) = -q) (i mod 5 = r) AND (i mod (-5) = r) (q * 5+r = i) AND (q * (-5)-r = -i) where i, q and r has the following values:

i	q	r
-15	-3	0
-14	-2	-4
-13	-2	-3
-12	-2	-2
-11	-2	-1
-10	-2	0
-9	-1	-4
-8	-1	-3
-7	-1	-2
-6	-1	-1
-5	-1	0
-4	0	-4
-3	0	-3
-2	0	-2
-1	0	-1
0	0	0
1	0	1
2	0	2
3	0	3
4	0	4
5	1	0
6	1	1
7	1	2
8	1	3
9	1	4
10	2	0
11	2	1
12	2	2
13	2	3
14	2	4
15	3	0

division:
mnemonic code: divinstr
trace No: 38

remainder of division:
mnemonic code: modinstr
trace No: 39

multiplication:
mnemonic code: mul
trace No: 40

2.1.10 NIL, Push, Pop

2.1.10

Nil-bit true is tested:
mnemonic code: nil
trace No: 66

Nil-bit false is tested:
mnemonic code: not nil
trace No: 67

The push and pop instructions are tested.

Test that r1 is nil after push:
mnemonic code: push
trace No: 68

Test that r1 is on top of stack:
mnemonic code: push
trace No: 69

Test that r1 is not nil after pop:
mnemonic code: pop
trace No: 70

Test that r1 is taken from top of stack:
mnemonic code: pop
trace No: 71

Test that r2 becomes the new top element:
mnemonic code: pop
trace No: 72

2.1.11 Set Instructions

It is tested that a variable is member of a set, when it has the value of one of the members in the set:
mnemonic code: settm
trace No: 41

It is tested that a set which is created can be empty or contains the expected values within its boundaries:
mnemonic code: setcr
trace No: 42

It is tested that sets can be adjusted from one number of members to another.
mnemonic code: setad
trace No: 43

It is tested that adjusted sets can be equal:
mnemonic code: seteq
trace No: 44

Set union is tested:
mnemonic code: setun
trace No: 45

Set intersection is tested
mnemonic code: setinter
trace No: 46

Subsets are tested:
mnemonic code: setsb
trace No: 47

Supersets are tested:
mnemonic code: setsp
trace No: 48

Set difference is tested:
mnemonic code: setdi
trace No: 49

2.1.12 Move Dataparts

It is tested that the instruction, which moves dataportions from one area to another, is performed.

Area of tree words is moved:
mnemonic code: moveg
trace No: 50

Area of tree bytes is moved:
mnemonic code: moveg
trace No: 51

2.2 Debug Console and ALU-Test

This part of the testprogram is testing the communication to the debug console by writing the whole readable set of characters on the console, and some selected ALU-functions (addition and carrybit). The ALU is tested to be able to do the following selected functions:

Addition of logical one:
mnemonic code: loc 0
trace No: 56

The carry shift from slice with bit 12-15 to slice with bit 8-11:
mnemonic code: carry 8-11
trace No: 57

The carry shift from slice with bit 8-11 to slice with bit 4-7:
mnemonic code: carry 4-7
trace No: 58

The carry shift from slice with bit 4-7 to slice with bit 0-3:
mnemonic code: carry 0-3
trace No: 59

All tree slice carries shifted at the same time:
mnemonic code: carry all
trace No: 60

Addition of 1 + (-1).

Carry bit = 0 in ps-register:
mnemonic code: carrybit 0
trace No: 61

Sum = 0:
mnemonic code: +
trace No: 62

Addition using MADD.

Carry bit = 1 in ps-register:
mnemonic code: carrybit 1
trace No: 63

Sum = 0.
mnemonic code: madd
trace No: 64

2.3 Scheduler and ALU-Test

2.3

In this part of the test 131 incarnations of a process are created and started. All free register-sets are used. All the processes uses SIGNAL, RETURN, and WAIT for communication. When finished the processes are removed.

3. TERMINATING THE TEST 3.

When once started, the test will run forever. Each time a run number is written on the console the test is completed.

When a termination is wanted, it must be done by addressing TOP, and type "KILL:CPU01"

Note : The only valid TOP-commands in relation with the CPU-test are:

```
NEW:CPU        ( starting the test )
KILL:CPU01     ( removing the test )
```

There will not be generated an errorstatistic as for other tests.

These specialities of the CPU-test are because of the need of simplicity of the test.

If started from OPSYS you must say something like:

```
>OPSYS
EX 47
BREAK CPU-C-00004
REMOVE CPU-C-00004
```

It is recomendable to break the test before removal to allow the test to release used registers.

4. TESTOUTPUT 4.

4.1 Normal Output 4.1

When the test is running in a CPU with no failures, it will generate an output like the following:

```
>TOP
NEW:CPU
cputest     initiated as cpu01
Select function:

>cpu-b-00001
-- CPU 200 test ---- ver 86.05.14 --
 !"£$%&´()*+,-./0123456789:;<=>? ABCDEFGHIJKLMN  ...
>cpu-c-00001
run no.          1
>cpu-b-00002
 !"£$%&´()*+,-./0123456789:;<=>? ABCDEFGHIJKLMN  ...
>cpu-c-00002
run no.          2
```

The letter after "cpu-" is the subtest. The number is run no. If some errors occurs an "e" is placed after the run no.

4.2 Error Output 4.2

The test has two different kinds of error-output.

The first one is telling that there is not enough memory to run the test.

"Creating of proc-022 result : 3 no free core."

The second one is telling about an instruction or a function that generated a wrong result.

```
**************** run no.     1  : ****************
cpu malfunction
mnemonic code :     stvsf.revsw
trace no : 35
**************************************************
```

5. TURN AROUND TIME 5.

The specified times are the approximate turn around times for each part of the CPU-TEST run in the RC3502/2 computer without any other programs running at the same time.

SUBTEST	TURN AROUND TIME
micro instruction test	30 s.
communication, ALU-test	15 s.
scheduler test	41 s.

A. REFERENCES

1. RCSL No. 99 0 00771
RC3502/2 Operating Guide

2. RCSL No. 30-M329:
RC3502, TOP35, Test Operating System,
User´s Guide

3. RCSL No. 52-AA1192:
RC3502/2 Reference Manual

4. RCSL No. 52-AA1177:
CPU212 - 219, Technical Manual

5. RCSL No. 99 0 00871
Debug Console Program for RC3502/2 Minicomputer

6. RCSL No. 52-AA1197
RC3502/2 Microprogram Listing

RETURN LETTER

Title: RC3502 CPU Testprogram Package, User's Guide

RCSL No.: 99 0 00913

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

_____

_____

_____

_____

Do you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

How can this manual be improved?

_____

_____

_____

_____

Other comments?

_____

_____

_____

_____

_____

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

PN: 99200176

..... Fold here .....

..... Do not tear - Fold here and staple .....

Affix
postage
here

REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark