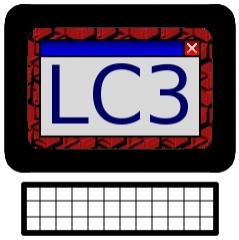
Complx

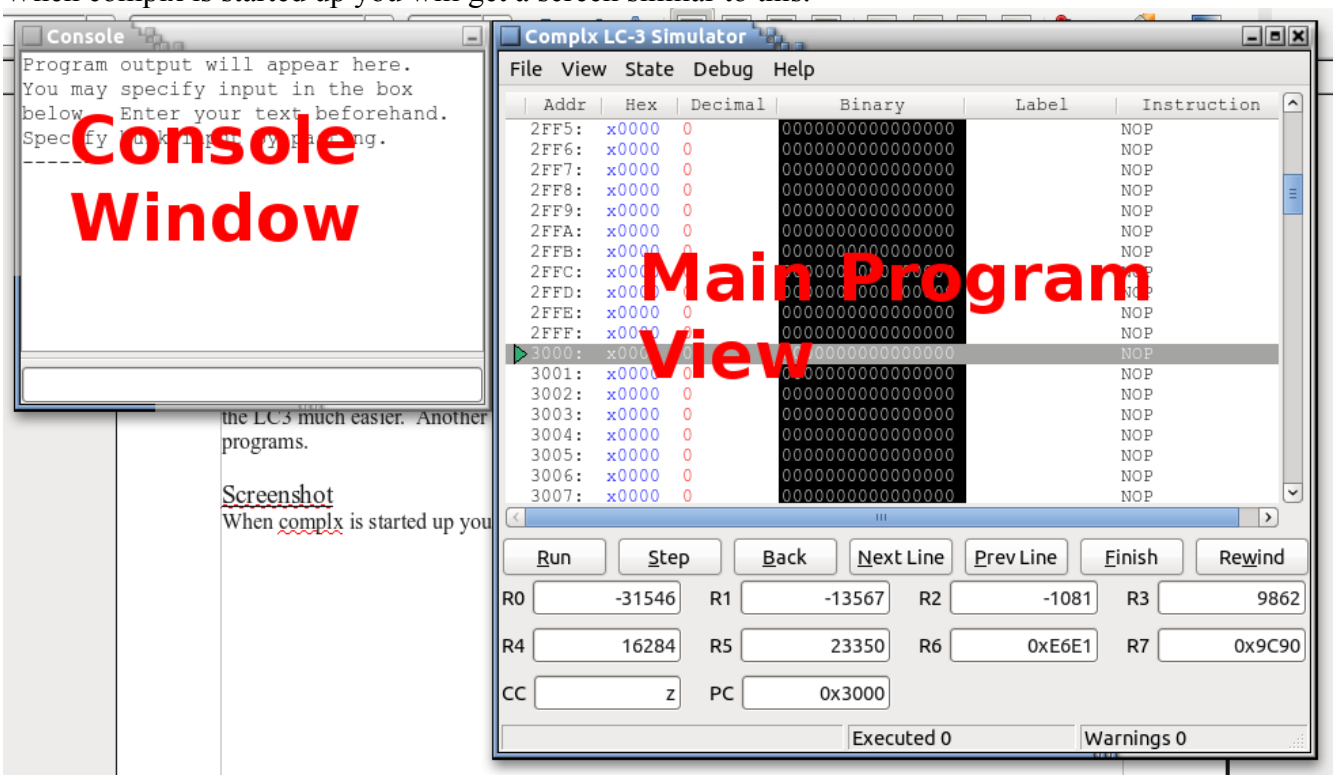


Introduction

Complx *pronounced Complex but spelled without the e* is an alternative to simpl (an old lc3 simulator). The goal of complx is to provide a simulator with more debugging features and make assembly programming with the LC3 much easier. Another goal of complx is to enforce the rules when writing assembly programs.

Screenshot

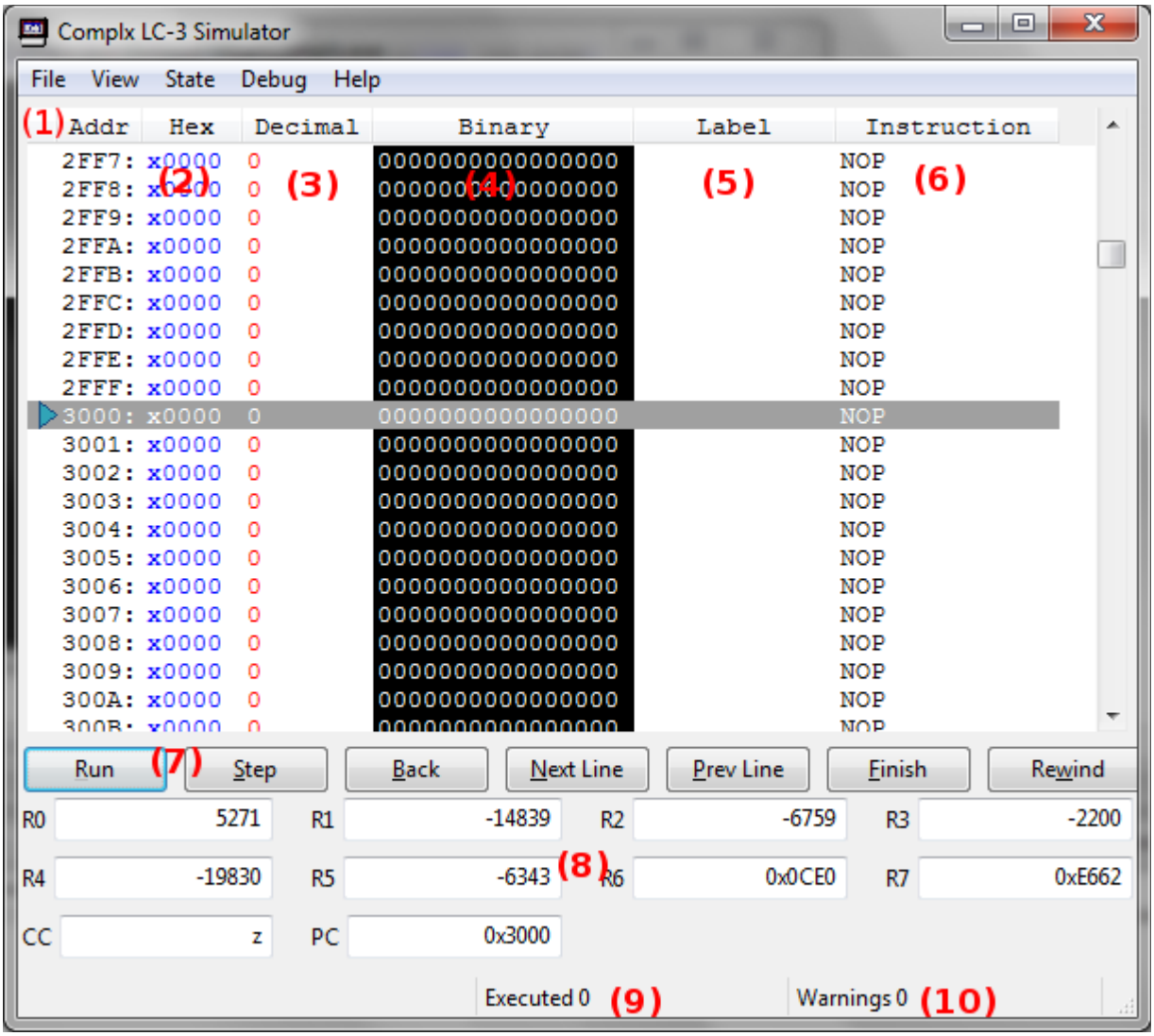
When complx is started up you will get a screen similar to this.



The console window allows the user to see the output of the program and any warnings their code generated. (Note: It may be minimized when the program starts)
The main program view shows a view of the LC3 memory the registers and the PC. It also allows the user to step through their program and change values/instructions in memory and registers on the fly.

How to use (A Walkthrough)

Shown below is the main view of Complx.



1. This is a view of the LC-3 memory you may see all of the addresses and what value it is in many bases. It also shows what symbol exists at that address and the disassembled instruction
2. For the LC-3 we express hexadecimal numbers preceding them with just an x (lowercase x). You may type hexadecimal numbers here and you will modify the address.
3. For the LC-3 we express decimal numbers preceding them with either # or nothing. I chose to display the decimal numbers without the # sign.
4. For the LC-3 there is no way (as far as I know) to express a number as binary however you should not need to use binary. However for complx you can express binary in this column.
5. Here you can type a symbol for the location.
6. The Instruction column shows you what the instruction disassembles to. You may change the level that complx disassembles to in menu View > Disassemble. You can also type an LC-3 instruction here and it will assemble it for you *Quick Fix.*

7. Here is the play area the buttons are explained as follows
    a. Run – Runs your code until it HALTs. If it appears your code is going into an infinite loop you can stop it by hitting the button again.
    b. Run For – Runs your code for at most X instructions, enter a negative number to undo at most X instructions
    c. Step – Runs only one instruction.
    d. Back – Undoes the actions of one instruction
    e. Next Line – Runs one line in the program (If the line happens to be a subroutine call then it will not step into it)
    f. Previous Line – Back steps one line in the program (If the line happens to be a subroutine call then it will not step into it)
    g. Finish – Finishes the current subroutine you are in (Step Out of the subroutine)
    h. Rewind – Undoes all of the instructions in the undo stack
8. Register View. Here you can see all of the registers including the CC register and the PC here. You can also edit the register's values on the fly.
    - Tip: Double click on the text box for a given register and you will change the base it displays.
    - Tip: you may also enter expressions when editing a registers value ex. R0 + R1 + PC – MEM[x3000] (Add R0 and R1 and the PC subtracted by what's in memory address x3000)
9. Executions. This shows you how many instructions your code executed before it HALTed
10. Warnings. This shows how many warnings your code generated (Your code must not produce any warnings).

Menus

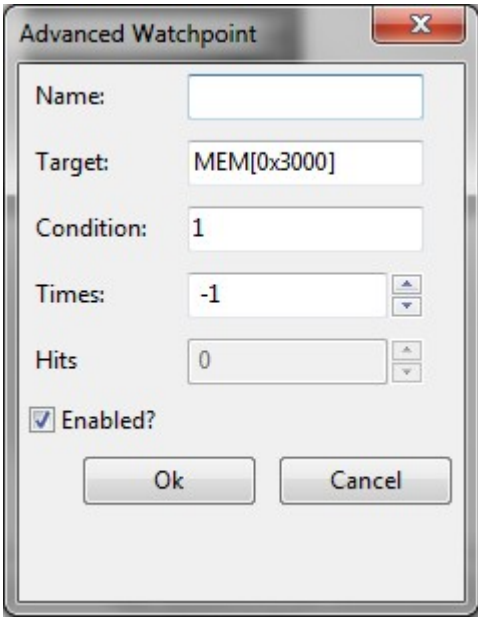
1. File Menu
    - Randomize and Load – Randomizes memory and loads an assembly file
    - Randomize and Reload – Randomizes memory are loads the last assembly file
    - Load – Loads an assembly file and reinitializes the machine.
    - Reload – Loads the last assembly file loaded and reinitializes the machine.
    - Load Over – Loads an assembly file over the machines current state.
    - Reload Over – Loads the last assembly file loaded over the machines current state.
    - Load Machine – Loads a machine state file.
    - Reload Machine – Reloads the last machine state file loaded.
    - Quit – Exits the program.
2. View Menu
    - New View – Creates a new memory view.
    - Goto Address – Jumps an address in memory
        - Tip: You can enter an expression
    - Disassemble – Edits the disassembling options
        - Basic – Does a straight disassemble without symbol table information.
        - Normal – Disassembles with symbol table information.
        - High Level – Disassembles into something C-Like.
    - Instruction Highlighting – Highlights different portions of the instruction.
    - Unsigned Decimal – Display decimal values as unsigned.
3. State
    - Control – Plays through or rewind instructions.
        - Step (F2) – Plays one instruction
        - Back (Shift + F2) – Undoes one instruction
        - Next Line (F3) – Plays one line (does not step into subroutines, traps, or interrupt handlers)
        - Previous Line (Shift + F3) – Undoes one line (does not step into subroutines, traps, or interrupt handlers)
        - Run (F4) – Runs until it halts
        - Run For (Ctrl + F4) – Runs for X instructions until it halts (asking everytime)
        - Run Again (Ctrl + Space) – Runs for X instructions until it halts (asking once)
        - Rewind (Shift + F4) – Undoes all instructions in the undo buffer
        - Finish (Shift + F5) – Steps out of a subroutine, trap, or interrupt handler
    - Randomize – Randomizes memory and registers.
    - Reinitialize – Resets memory to an initial state.
    - True Traps – Enables true trap handling.
    - Interrupts – Master interrupt switch allows devices to generate interrupts
    - Clear Console – Clears the LC-3 console.
    - Clear Console Input – Clears all of the input waiting in the console
4. Debug
    - Undo Stack – Sets the undo stack, how many instructions you can go back. (The default

is 65536 instructions).
    - Breakpoints and Watchpoints – Views all break points and watch points entered.
    - Add Temporary Breakpoint – Adds a one time breakpoint.
    - Add Breakpoint – Adds a permanent breakpoint.
    - Add Watchpoint – Adds a watch point on a memory address or register.
    - Add Advanced Breakpoint – Adds a conditional breakpoint.
    - Add Blackbox – Adds a blackbox (you will never step into something thats blackbox'd)
5. Test
    - Run Tests – Runs an lc3test xml file
    - Rerun Tests – Runs the last test file to be loaded
6. Help
    - Documentation – Displays this file you are reading.
    - LC-3 ISA – Displays the LC-3 ISA document.
    - About – Displays information about this application.

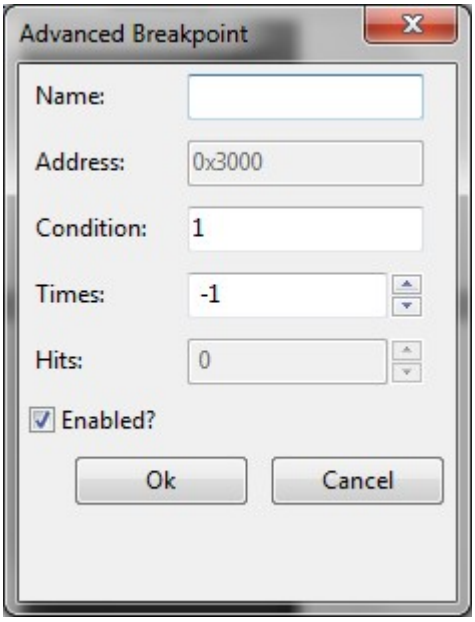
Debugging

The debugging menu is Complx's prime feature. It allows the user to set various breakpoints and watchpoints in memory so that they can find a bug more easily. Here are the menu options and relevant menus and how to effectively use them.

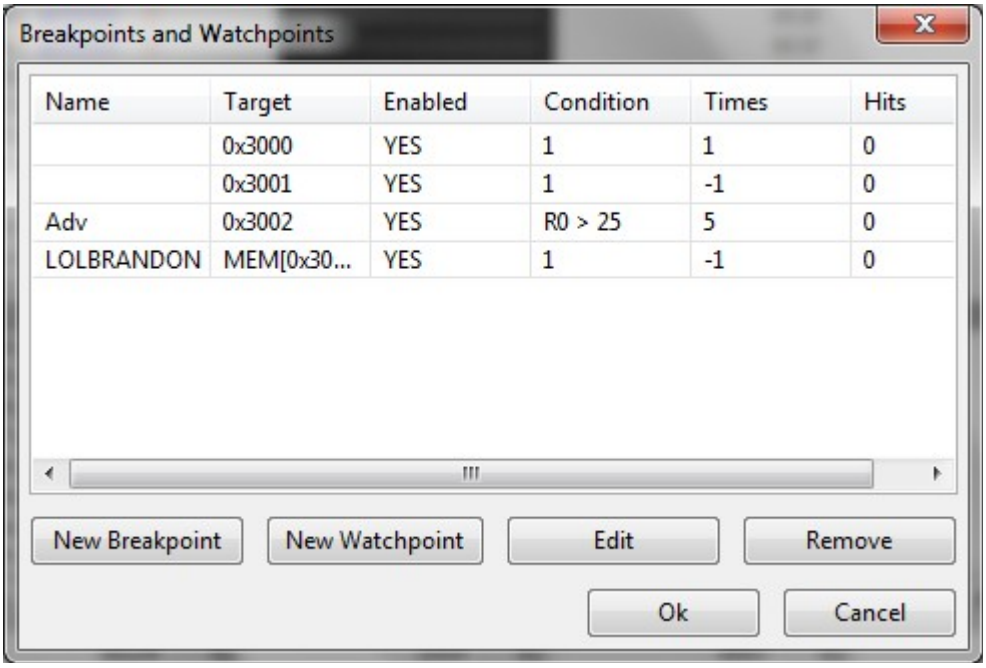
1. Add Temporary Breakpoint – Shortcut F8
   a. Adds a temporary breakpoint what this means is that when the breakpoint has been triggered it disappears. This is useful if you want to run code until a certain point in your program.
2. Add Breakpoint – Shortcut F9
   a. Adds a breakpoint unlike the temporary breakpoint once it has been triggered it does not disappear. When the code reaches this point it will stop your code.
3. Add Blackbox – Shortcut F5
   a. You can add a blackbox to a subroutine, trap, or interrupt handler here. A blackbox will automatically completely execute the thing that is blackbox'd when it is encountered.
4. Add Watchpoint – Shortcut F11
   a. Adds a watchpoint. A watchpoint is a special type of breakpoint that activates only when a value is written to memory, in this case memory or a register.
   b. Parameters are as follows:
      i. **Name** – Watchpoint name. Has no effect on its operation
      ii. **Target** – Watchpoint target. Can be an address or a register.
      iii. **Condition** – A Boolean expression that is evaluated when the target is written to. If the result is 0 it does not activate the watchpoint if the result is not 0 then the watchpoint activates and stops execution. Example MEM[0x3000] > 25.
      iv. **Times** – The number of times the watchpoint needs to be triggered before it is deleted. (-1 means permanent)
      v. **Hits** – Lets you know how many times the watchpoint has been activated.
      vi. **Enabled?** – Disables or enables the watchpoint.



5. Add Advanced Breakpoint – Shortcut F12
   a. Adds an advanced breakpoint. You can modify many different parameters of the breakpoint here.
   b. Parameters are as follows:
      i. **Name** – Breakpoint name. Has no effect on its operation.
      ii. **Address** – Address where the breakpoint sits.
      iii. **Condition** – A Boolean expression that is evaluated when the code reaches the breakpoint. If the result is 0 it does not activate the breakpoint if the result is not 0 then the breakpoint activates and stops execution. Example R0 > 25.
      iv. **Times** – The number of times the breakpoint needs to be triggered before it is deleted. (-1 means permanent).
      v. **Hits –** Lets you know how many times the breakpoint has been activated.
      vi. **Enabled –** Disables or enables the breakpoint.

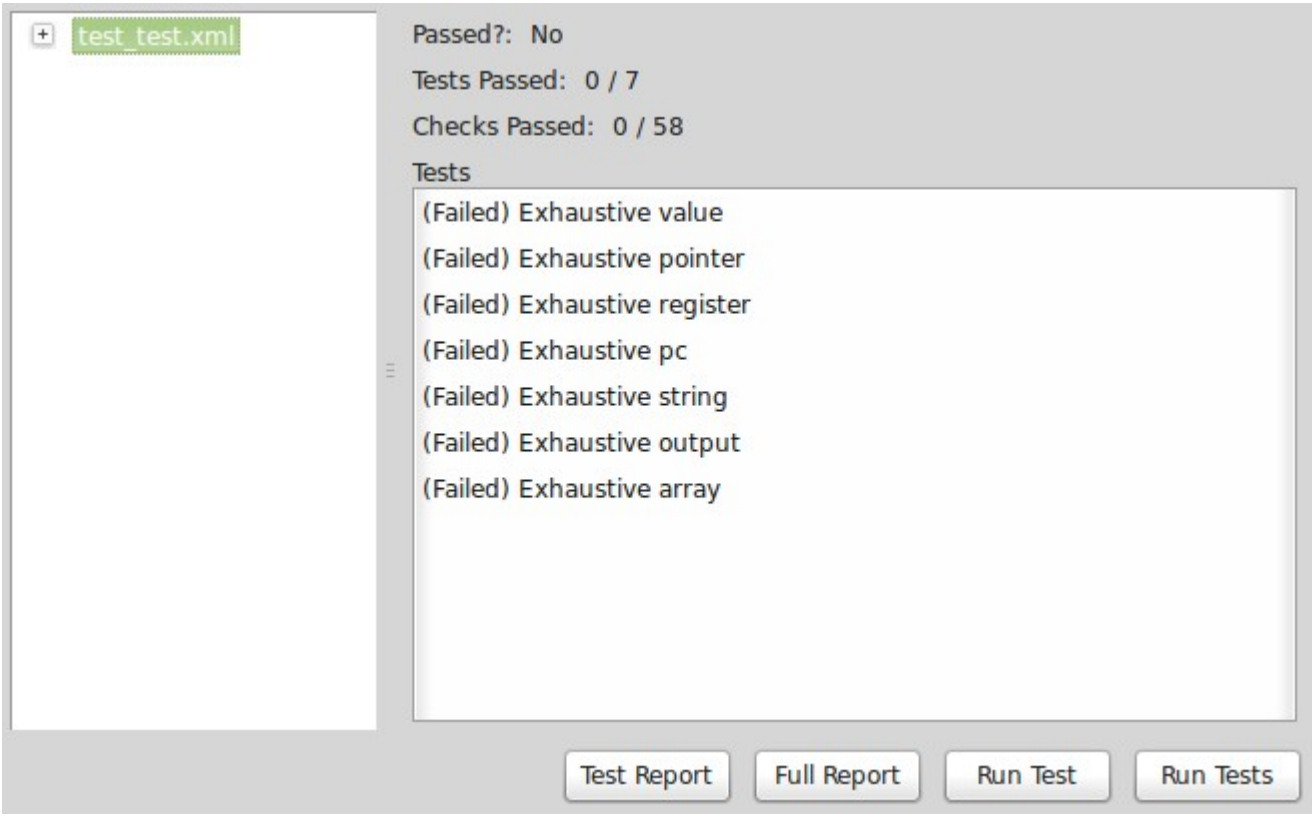


6. Breakpoints and Watchpoints – Shortcut F7
    a. Views all breakpoints and watchpoints that are currently created
    b. Here you can add additional breakpoints/watchpoints, edit, or delete them.



Testing

The testing menu was added as of Complx 4. This menu has similar gui based functionality as the textual based program lc3test introduced October 2012. This menu can be accessed by selecting Run Tests and giving it an xml file that contains the test cases. Afterward you will get a screen similar to this.



In the left widget a control is present to show you all of the tests and conditions that are present within the test file in a hierarchical structure. To the right is a panel that shows information about the item that is selected from the left control.

Definitions you should know is that there are three things you can select from the tree structure to your left.
A **Test Suite** which contains many tests.
A **Test** which specifies the pre and post conditions for the test and other features about how the state of the lc3 should be when the test is ran.
And **Postconditions** which are conditions which must be true after the test has completed.

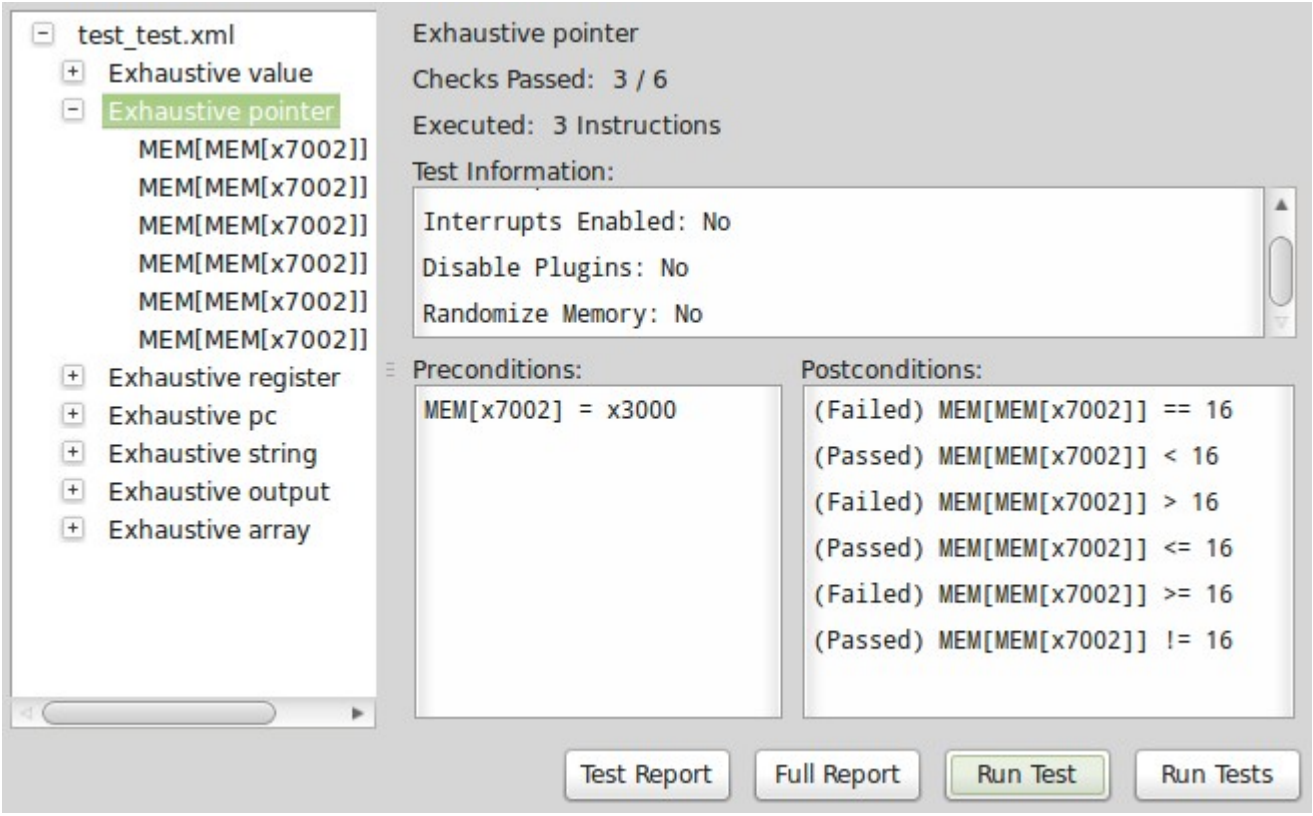
The buttons on the bottom do the following

1. Test Report. Gives a report about how a particular test went. Note that a test must be selected in order for the report to be generated.
2. Full Report. Gives a report about the entire Test Suite.
3. Run Test. Runs a single test. Note that a test must be selected for this to work
4. Run Tests. Runs all of the tests in the test suite.

Here since the file (test_test.xml, a Test Suite) was selected it will show information about the stats about everything in the test suite:

1. Whether or not you passed the entire test suite.
2. The number of tests you completely passed
3. The number of post conditions you have passed (as a total)
4. And all of the tests associated with the test suite.

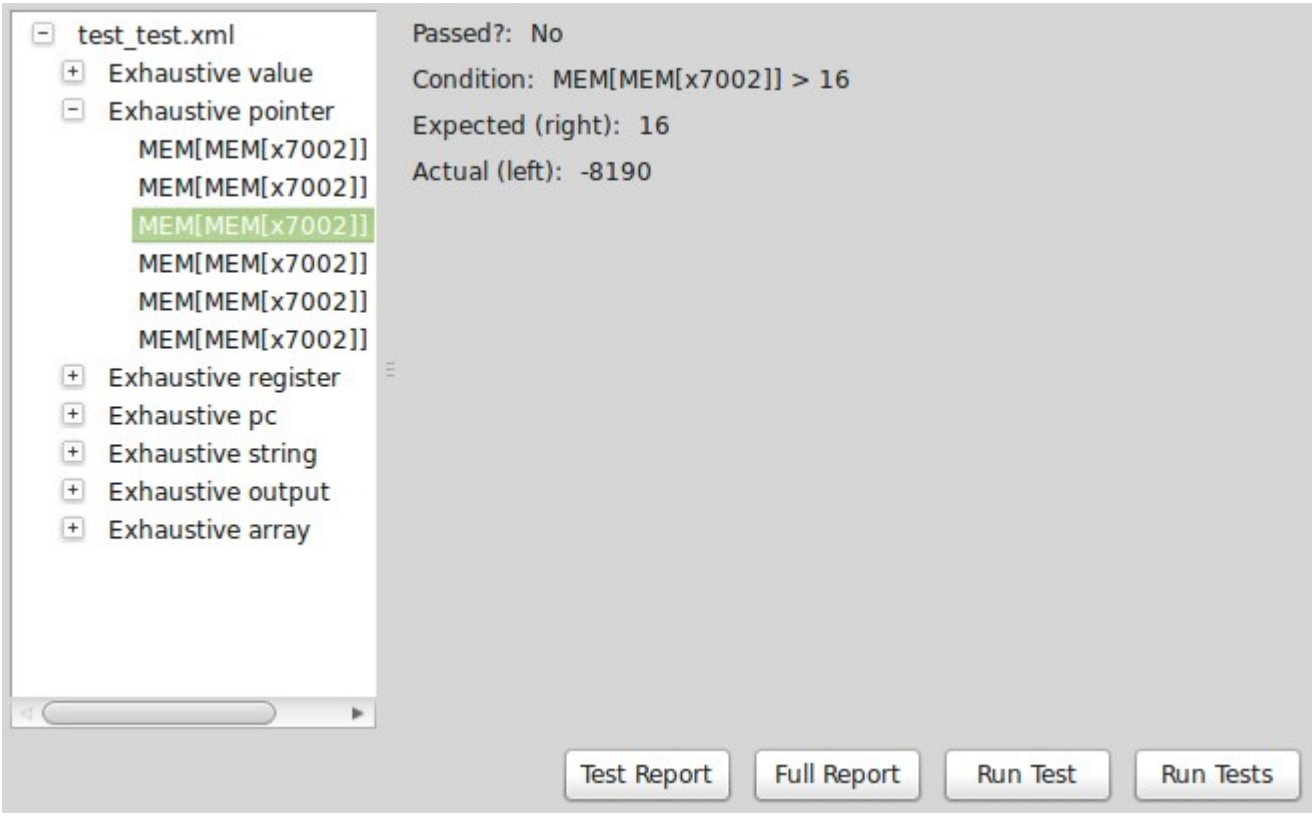
When a test is selected the screen on the right changes to show the stats for the test that was selected. You should get a screen similar to the following.



Here the following is displayed

1. The name of the test case Exhaustive pointer in this case.
2. The number of post condition checks that passed
3. The number of instructions that was executed before the machine halted.
4. Parameters about the states environment that will be true when the test is ran.
5. The preconditions of this test. Memory address x7002 will be set to x3000.
6. The post conditions that will be checked after the test is ran.

When a post condition is selected from the control on the left the screen on the right changes to show stats about the post condition.



Here the following information is displayed

1. Whether or not you've passed the post condition.
2. A textual string representing the condition that needs to be passed.
3. The expected value or right hand side of the condition.
4. The actual value your code produced or the left hand side of the condition.