ARM Support Functions and ARMSim#2.0

In the folder Resources/Assignment 4/Supplied Code, there are two new files:

- CbRuntime.s
- TestCbRunTime.s

And this on-line zipfile

https://dl.dropbox.com/u/43861167/ARMSim.zip

contains a Windows installer for ARMSim2.0 (which is not the same version as the one used in csc230 at UVic).

After installing, you can check that ARMSim works by running the following test.

1. Start ARMSim (it may be a bit slow to start the first time -- be patient)
2. Open the File/Preferences menu and on the Plugins tab, make sure that the plugin named 'AngelSWIInstructions' is checked. You may also want to make sure that both 'Protect Text Area' and 'Stop program in misaligned memory access' are checked on the 'Main memory' tab. Then click OK in the bottom right corner. (These preferences are remembered for future invocations of ARMSim.)
3. Open the File/Load Multiple menu; use the Add button to select TestCbRunTime.s and then use it again to select CbRuntime.s. Both files should now be listed in the main pane. Click the OK button. This causes the two files to be assembled and translated to their '.o' equivalents.
4. Both source code files should now appear as separate tabs in the main pane of ARMSim. There should also be a separate pane named 'OutputView' with two tabs named Console and stdin/stdout/stderr. If the pane is not visible, open the View options on the ARMSim menu bar and make sure that 'Output' is checked.
5. Either select Debug/Run or click the right pointing blue triangle to run the program.
6. The stdin/stdout/stderr tab of OutputView should now contain text generated by the program. The last line displayed is a prompt for you to enter a series of integers, ending with a zero. Enter a few integers and verify that they are echoed correctly.
7. When your CFlat compiler is working, it should generate a '.s' file which can be combined with CbRuntime.s and be executed in a similar manner.

Do read through TestCbRunTime.s because it gives an idea of the style of code which can be generated by your CFlat compiler. (There is no need to study CbRuntime.s – it uses a rich repertoire of ARM instructions, beyond that which would be generated by the CFlat compiler, and much of the code is interfacing with support functions provided by the Angel SWI plugin.)