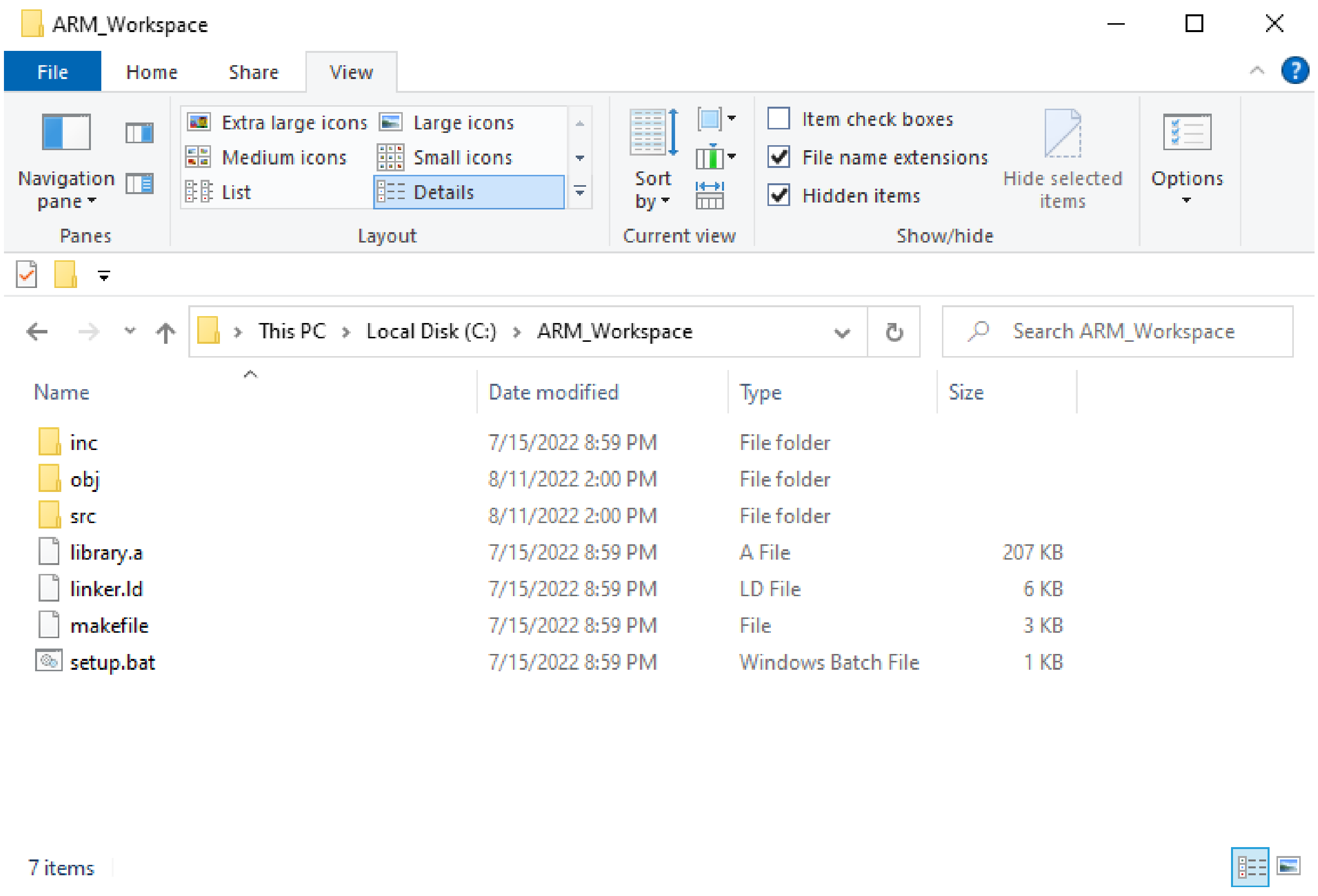
Lab #1B – Example ARM Calculator Program

Goals:

Learn how to use the GNU programming tool chain for building ARM programs
Learn how to use the debugging window in the Discovery Board library

Step 1: Download the "Using the GNU Toolchain.pdf" document posted in the first module in Canvas to your laptop, and refer to it through this lab exercise as a supplementary reference. Note that this toolchain has already been installed for you on the lab computers, but if you install and use this toolchain on your personal computer as well, this document provides a guide for various operating systems.

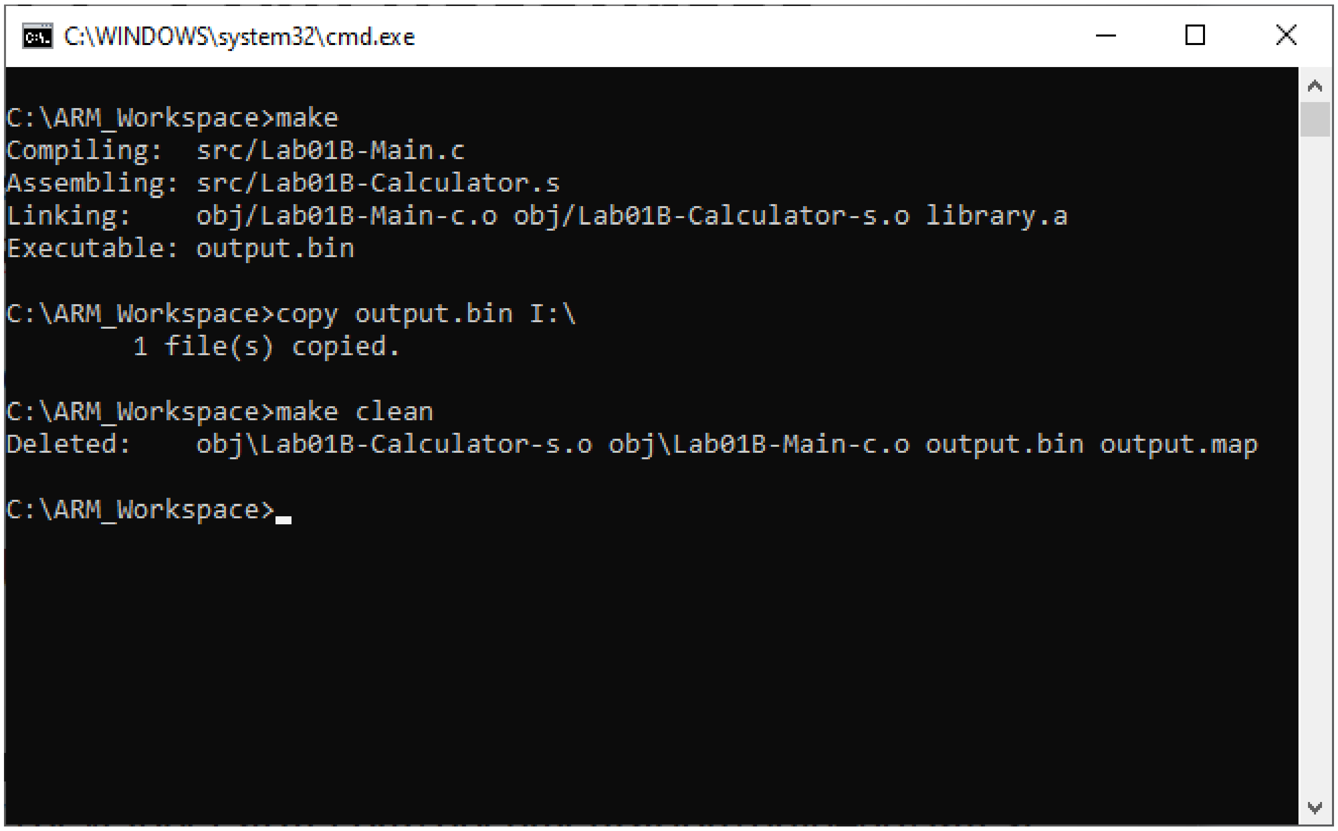
Step 2: Download the ZIP archive file 'ARMProgrammingWorkspace.zip' from the first module in Canvas to the lab computer. If it has not already been created, create a new workspace folder on the lab computer for building your ARM programs at a convenient location, such as C:\Workspace for example. Unzip the 'ARMProgrammingWorkspace.zip' archive into this folder. The folder contents, as viewed in Windows Explorer, should look like



Name	Date modified	Type	Size
inc	7/15/2022 8:59 PM	File folder	
obj	8/11/2022 2:00 PM	File folder	
src	8/11/2022 2:00 PM	File folder	
library.a	7/15/2022 8:59 PM	A File	207 KB
linker.ld	7/15/2022 8:59 PM	LD File	6 KB
makefile	7/15/2022 8:59 PM	File	3 KB
setup.bat	7/15/2022 8:59 PM	Windows Batch File	1 KB

Step 3: Now start building your example program. Download the two source code files Lab01B-Calculator.s and Lab01B-Main.c from the Week #1 module in Canvas to the 'src' folder of your workspace. (These are example source code files from the Dan Lewis website.) Make sure that these are the only source files in the 'src' folder at this point.

Step 4: In the Windows file manager, double-click on the file displayed as either ‘setup’ or ‘setup.bat’ (depending on your file display options) in your workspace. This should pop up a command prompt window on your screen, with various command search paths set up for you. Type the simple command ‘make’ at the command prompt and check if the code builds with no errors. A new binary executable file ‘output.bin’ should now appear in your workspace folder.



```
C:\ARM_Workspace>make
Compiling:  src/Lab01B-Main.c
Assembling: src/Lab01B-Calculator.s
Linking:    obj/Lab01B-Main-c.o obj/Lab01B-Calculator-s.o library.a
Executable: output.bin

C:\ARM_Workspace>copy output.bin I:\
        1 file(s) copied.

C:\ARM_Workspace>make clean
Deleted:    obj\Lab01B-Calculator-s.o obj\Lab01B-Main-c.o output.bin output.map

C:\ARM_Workspace>_
```

Step 5: Plug in the STM32F429I-DISC1 Discovery board into a USB port on the lab computer. This should create a new Windows device drive, such as ‘E:\’ or ‘F:\’, in the lab computer file system. Copy the ‘output.bin’ executable file into the top directory of the new drive. You can either use a command line ‘copy’ command as shown above (but copying to whatever Windows device gets assigned to your Discovery board, not necessarily I:\) or copy the file with point-and-click operations in the File Manager. Watch the red and green LEDs alternating to indicate the file is being installed on the ARM processor on the board.

Step 6: Test your calculator application with several example calculations in all three modes: Decimal, Hex, and Binary. Type your calculations into the text area for this Canvas assignment. Demonstrate your calculator application working to the lab TA.

Step 7: On the lab computer, open the Visual Studio Code programming editor, or another editor of your choice, and edit the Lab01B-Calculator.s source code file in the ‘src’ directory. Locate the section of the assembly code that performs an addition operation. This function is called by the main calculator program when you press the ‘=’ key after entering an addition operation. Insert one new source code line so that the function now reads:

```
          .global         Addition
          .thumb_func
          .align
Addition: // R0 = op1, R1 = op2                   Insert this new line
          SVC             #0
          ADD             R0,R0,R1      // R0 = R0 + R1
          BX              LR
```

This new line inserts a debugging breakpoint into the assembly code. When execution reaches this new statement, the program is suspended and a yellow debugging window will be popped up on the screen displaying the hex contents of all the core registers in the ARM processors.

Step 8: Rebuild the program by entering the ‘make’ command in the command window and download the updated ‘output.bin’ file into the Discovery board.

Step 9: Put the calculator application into hex mode and enter the keystrokes to add two examples hex numbers. Note the debug window now popping up when you hit the ‘=’ key. Type into the text area for this assignment the calculation you entered, the hex contents displayed for registers R0 and R1, and note any similarities in the assignment text area. Also enter the program memory location of the SVC instruction that triggered the debug breakpoint. This is displayed on the top line of the yellow debug window. Demonstrate your debug window working to the lab TA. Finally press the blue User button on the Discovery board to release the breakpoint and resume program execution. Note the calculation finishing on the screen. *Keep in mind this handy debugging facility in the board library available to you as you develop and debug your assembly code this semester!*

Step 10: Clean the programming workspace by typing the command ‘make clean’ at the command prompt, and by deleting the source code files in the ‘src’ folder. The workspace is now ready to be used for building another program.