USING THE GNU TOOLCHAIN

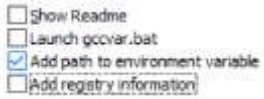
IMPORTANT: (1) Never use a *word processor* like MS/Word, WordPad or Apple's TextEdit to edit any file in COEN 20, since doing so will often introduce invisible formatting codes that cause syntax errors. Suitable editors include Window's Notepad and Apple's Script Editor. (2) Never use filenames containing spaces or filename extensions with uppercase characters (e.g., use foo.c instead of foo.C).

STEP 1: INSTALLING THE GNU TOOLCHAIN SOFTWARE

Use a browser to open the textbook website located at http://www.engr.scu.edu/~dlewis/book3/. In the "Software" section, click on "Workspace for the GNU ARM Embedded Toolchain" to download the file `workspace.zip`, and unzip it into a new workspace folder on the desktop.

Microsoft Windows:

1. Click on "GNU ARM Embedded Toolchain …" to open its download page. Locate and download the Windows version of the toolchain. Use all the default settings during the install procedure until the very last screen. On that screen, uncheck the three boxes that are already checked by default and instead check the third box labelled "Add path to environment variable":

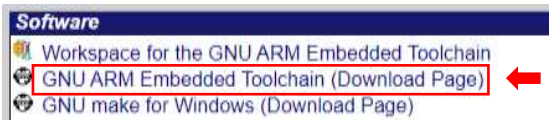


2. Click on "GNU make for Windows …" to open its download page. Locate `GnuWin32` (a make program for Windows). Download the file described as "Complete package, except sources". Install it using the default settings.

3. Repeat step 2 for "GNU grep for Windows…"

Apple Mac or Linux:

1. Click on "GNU ARM Embedded Toolchain …" to open its download page. Locate and download the appropriate Linux or Mac OS X version of the toolchain and install it using the defaults. (Select the .pkg file for a Mac.)



2. Locate the file `'makefile'` in your workspace folder. Edit the definition of symbol `PREFIX` to be the name of the bin subdirectory within the directory where you installed the toolchain. For example, if the tools were installed in the default installation directory named `/Applications/ARM`, then the edited line should read `PREFIX=/Applications/ARM/bin/` (the trailing slash is mandatory).

3. You may delete the file `'setup.bat'` in your workspace folder; it's only used on Windows.

STEP 2: BUILDING YOUR FIRST PROGRAM

1. In the "Course Materials" section of the textbook website, hover over "Programming Labs" and click on "Lab1A: 16-bit Calculator". When the PDF document opens, locate the yellow download icons in the upper right corner. Right-click and use your browser's "Save Link As" option to download the files `Lab1A-Main.c` and `Lab1A-Calculator.s` to the `src` subdirectory of your workspace folder.

2. ***Windows Only:*** Double-click on `setup.bat` in the root of the workspace folder. This will open a command line window with the root of the workspace folder as the current directory.

   Apple Mac and Linux Only: Open a terminal window and change the current directory to the root of the workspace directory.

3. Enter the command `'make'`. This will run the compiler, the assembler and the linker, producing the final program in a file named `output.bin` located in the workspace folder.

   NOTE: The make program accepts a command line parameter to specify other functions:

`make clean`	Removes all `*.o` files in the `obj` subdirectory, and `output.*` files in the workspace directory. (Used before rebuilding a program from scratch using files already in the `src` subdirectory.)

USING THE GNU TOOLCHAIN

`make cleanall` Same as `make clean`, but also removes all `*.c` and `*.s` files in the `src` subdirectory. (Used to remove any left-over files before starting a new lab assignment.)

➡ ***Important Notes for Apple Mac Users:***

The more recent versions of the Mac operating system do not include the make program by default. If you get an error when you try to run make from a terminal window, you may need to do a manual install by entering this command in the window:

```
xcode-select --install
```

Depending on your security settings, you may not be able to run `'make'` because components of the GNU Toolchain are "from an unidentified developer". In order to fix this, enter the following command in your Terminal window:

```
sudo spctl --master-disable
```

Note that this will allow your computer to open applications downloaded from anywhere. To change your security settings back to default (which only allows you to open applications from the App Store and identified developers), enter the following command in your Terminal window:

```
sudo spctl --master-enable
```

STEP 3: DOWNLOADING YOUR PROGRAM TO THE BOARD

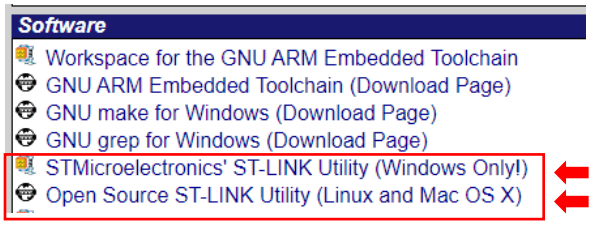
The procedure for downloading a program depends on the version of the board as identified by a model number (DISC1 or DISC0) located just above the display.

STM32F429I-DISC1 (with STLINKv2 Firmware):

Plug one end of a mini-USB cable into the larger USB connector at the top of the board and the other end into a USB port on your computer. A window on your computer should open that displays a list of files that reside on the board. Locate the file `output.bin` in your workspace folder and copy it to the root directory of the board. The board will automatically load and execute the code.

STM32F429I-DISCO (with STLINKv1 Firmware):

Plug one end of a mini-USB cable into the larger USB connector at the top of the board and the other end into a USB port on your computer. Then follow the instructions below to install a small utility from the textbook website and use it to download your program to the older DISC0 board.



Windows: Click on "STMicroelectronics ST-LINK Utility" to download the STMicroelectronics ST-Link Utility program. Once unzipped and installed, use the File menu to open the `output.bin` file you created. Next, use the "Program…" option on the Target menu to download the program to the board. A "Download" window will pop-up; check that the "Start address" is `0x08000000` and then click "Start".

Apple Mac and Linux: Click on "Open Source ST-LINK Utility" to download the open source Linux or Apple Mac version of a program called `st-flash`. Once installed, you can use it to download programs to the STM32F429 development board. Copy the file `output.bin` to the directory where you installed the open source version of `stlink` and use the following command line:

```
st-flash write output.bin 0x8000000
```

DEBUGGING YOUR ASSEMBLY LANGUAGE SOURCE CODE

The ARM assembly language instruction `SVC` and its single operand (an integer constant between 0 and 255) provide a convenient debugging tool. You can insert it into your assembly language code at any point to see the current contents of the registers. Pressing the blue pushbutton removes the information from the display and continues execution of your program.

`SVC 0`: Displays the current contents of CPU registers R0-R12 and the flags NZCV.
`SVC 1`: Displays the current contents of floating-point registers S0-S15.
`SVC 2`: Displays the current contents of floating-point registers S16-S31.
`SVC #`: Displays simply "`SVC #`", where '#' is an integer between 3 and 255.