

## Creating and Simulate an ASM Project in CCS

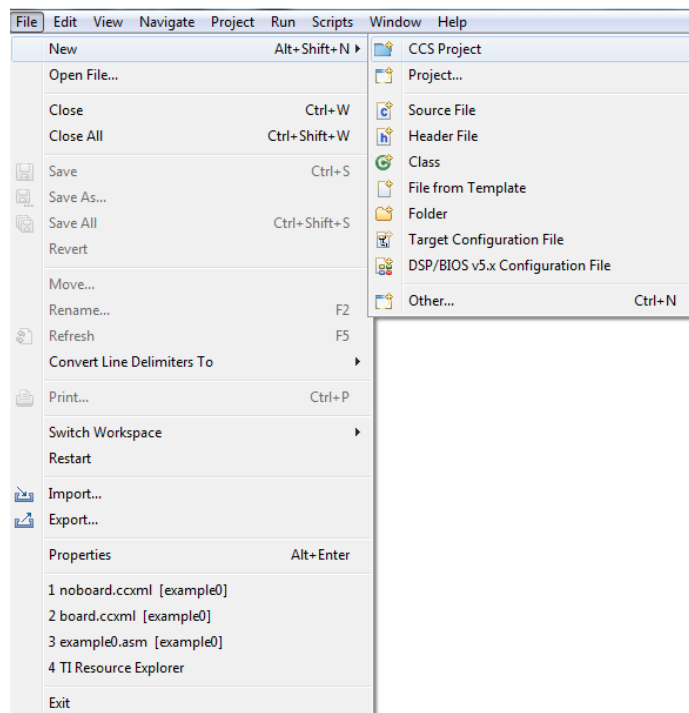
### Introduction

The purpose of this document is to enable a student to quickly create a project under CCS for assembling, and linking an assembly file, and then to simulate the file. To complete this tutorial you will need two additional files: **ex0.asm** and **KG\_RAM\_Link1.cmd**. After obtaining them from our class web page, copy the **KG\_RAM\_Link1.cmd** file into a folder called **c:\4744\ccs\projects\example0** and put the **ex0.asm** file on your desktop (or in the same folder).

The file **ex0.asm** is an assembly file that illustrates various assembler directives and TMS320F28335 DSP assembly code. **KG\_RAM\_Link1.cmd** is a linker command file that is required to set the proper addresses for each code section and to instruct the linker how to build the final machine code output. This linker command file is an adaptation of Texas Instrument’s linker command file “28335\_RAM\_Ink.cmd.”

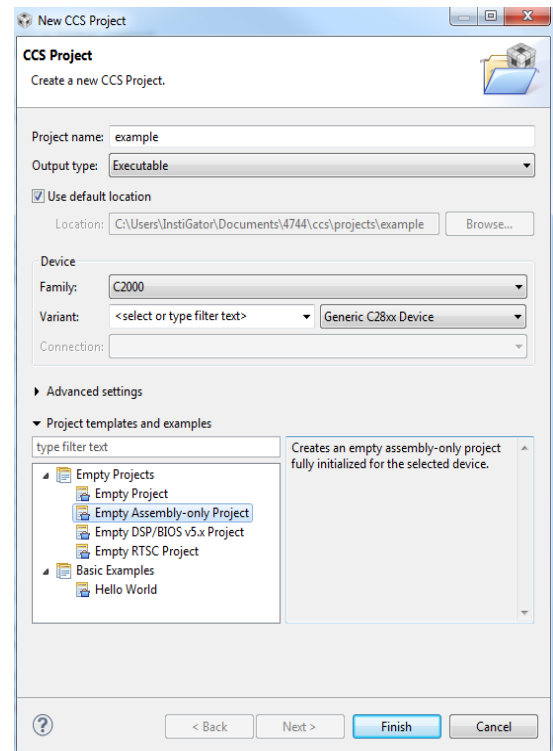
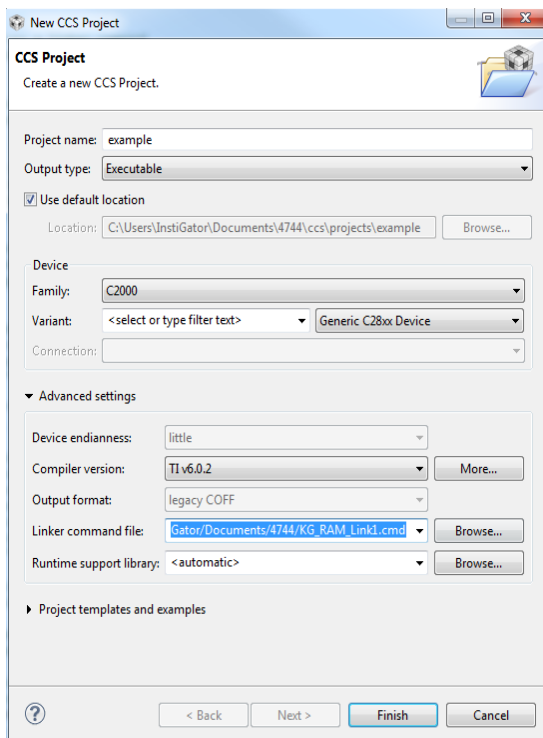
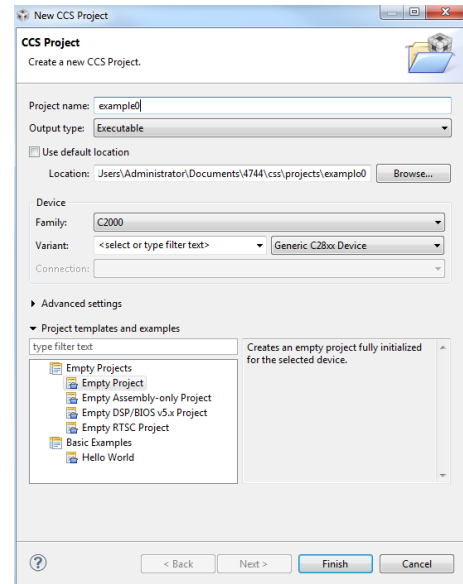
### Procedure

1. This tutorial assumes that you already have Code Composer Studio installed and have set your workspace folder location on your hard disk. For more information on this topic see the tutorial **CCS\_Installation\_Instructions.pdf** (at [http://mil.ufl.edu/4744/docs/CCS\\_Installation\\_Instructions.pdf](http://mil.ufl.edu/4744/docs/CCS_Installation_Instructions.pdf)).
2. Open CCS and create a new project via the following commands (as shown below): **File → New → CCS Project**.



## Creating and Simulate an ASM Project in CCS

- Type in your Project Name, e.g., **example0**. Next, **un-check** the box “Use default location” to place the project in another location other than your workspace folder. It is recommended that you create a new folder called **projects** underneath the folder you created in the CCS installation tutorial, CCS\_Installation\_Instructions.pdf. In this folder you should then create a new folder for every project, e.g., **c:\4744\ccs\projects\example0**. Note: CCS does not create a specific file to contain the project information but instead looks for files under a particular name in a given folder. This will be discussed further later; the important rule is to create a folder for every new project (examples, lab code, experimentation, etc.). See the screen snapshot to the right. Select “Next >”.
- Select the ‘Advanced settings’ tab. This should drop down a new menu for you to select options. Click the drop down box for the Linker command file. Point to the KG\_RAM\_Link1.cmd file. This file is located on the website and in the example0.zip file. You will use this linker file for the entire semester. Next, select the ‘Project templates and examples’ tab and then highlight the Empty Assembly-only Project. Click finish. See the screenshots below.

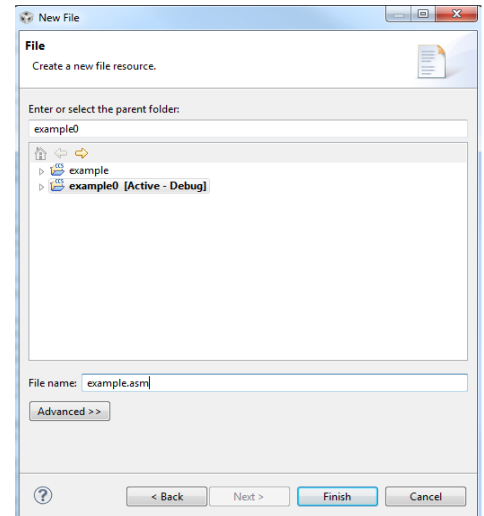


## Creating and Simulate an ASM Project in CCS

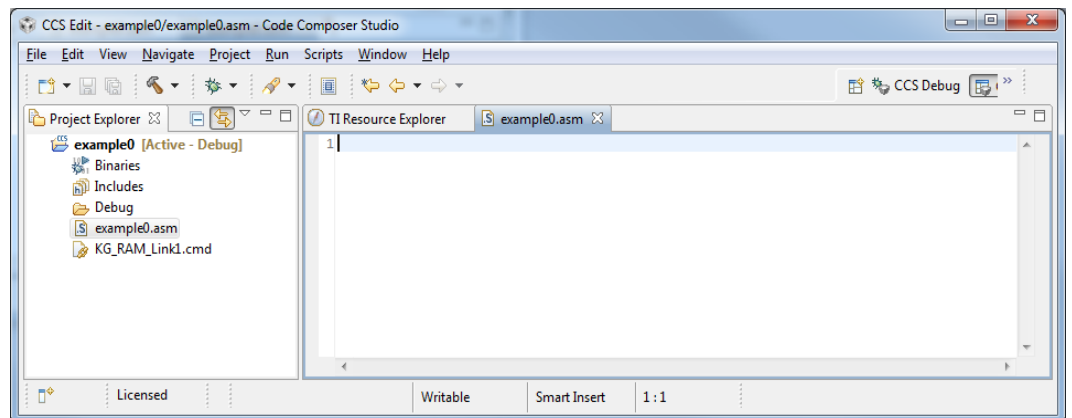
We now have created the project dependencies (libraries to be used for assembly and memory map for the machine code output) and will now create the actual assembly file.

- To create a new ASM file press: Right click the project in the Project Explorer window and then go to **New** → **File**. This can also be done by selecting **File** → **New** → **Other** → **General** and then selecting File. For this example, I used the filename example0.asm.

**Note:** Make sure to write the .asm extension following your file name. Failure to do this will generate an error when trying to compile your code. If you type the .asm extension correctly, a line number “1” will appear next to the cursor in the new file (window) as show below.



- You should now see this created project. Note the included files in the left most window. The new asm window has a “1” indicating line number. At this point you can type in new ASM code in the center window. However, since you probably are not proficient at writing F28335 DSP assembly yet, you should copy

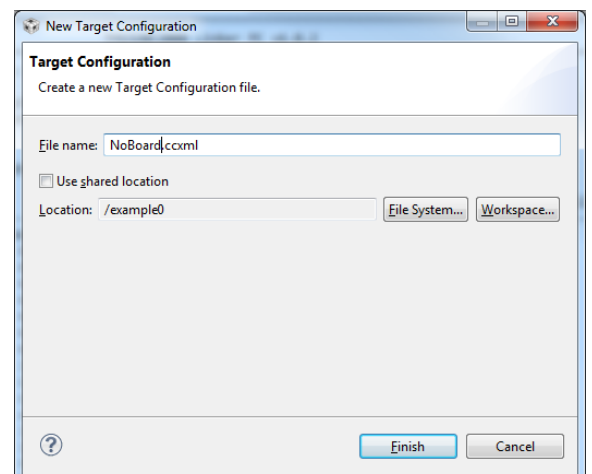


in the assembly code from our example code file that you previously copied to your desktop: **ex0.asm**. To do this press **File** → **Open** **File** → and then select **ex0.asm** from the directory where you originally placed the file. When **ex0.asm** is open, copy the contents using **Edit** → **Select All** and then **Edit** → **Copy** in the **ex0.asm** window to copy the code to your new **ex0.asm** window with **Edit** → **Paste**

- To generate a list file (a file with a .lst extension, which will appear in the Debug folder) contains both the assembly language file, as well as machine codes, line numbers, and other useful information) open the .asm file and then go to **File** → **Properties** and then select **Assembler Options**. Then select (i.e., check) “Generate listing file (--asm\_listing, -al)”. While there, also select “Generate asm extended warnings (--asm\_remarks, -mw)”.

- We are now almost ready to compile the code but must create a target configuration file to tell CCS what simulator or emulator/programmer you will be using for the code.

- Create a new Target Configuration File by pressing **File** → **New** → **Target Configuration**. The target

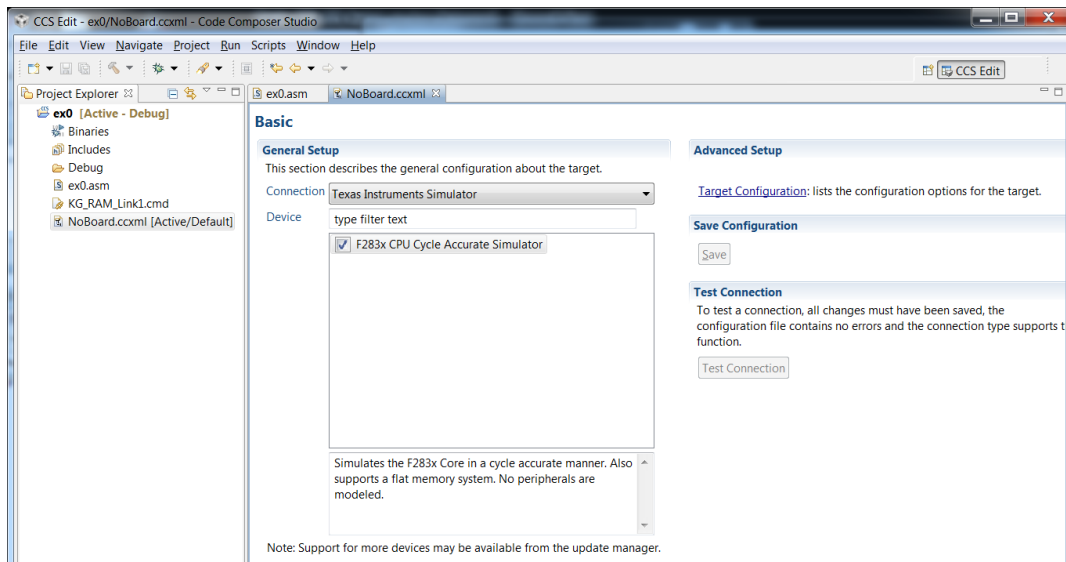



## Creating and Simulate an ASM Project in CCS

configuration file will identify the simulator or type of emulator CCS will use for programming and emulating the DSP board, i.e., it tells CCS where to load the program. In our case the simulator is in CCS and the programmer/emulator hardware is actually on our lab board.

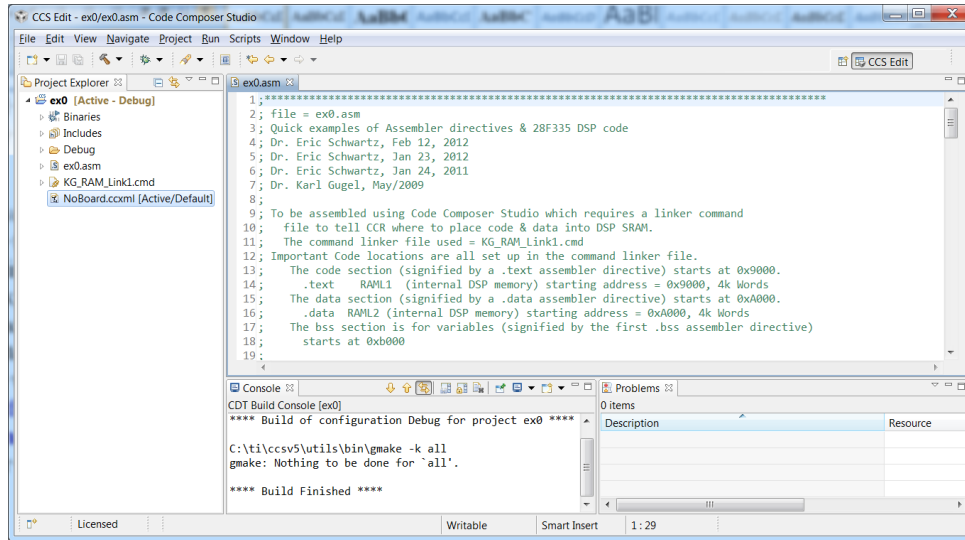
Type in a filename **NoBoard** to remind us this target configuration is for simulation only; the ccxml file extension is automatically added. Next, un-check the **Use shared location** box and then press **Workspace** to select the current project location. If the current project location shows up in the “Location:” field (i.e., /example0), you can leave **Use shared location** checked and just press **Finish**. We want to make sure we add the target configuration file to our current project files. Click **Finish** after the location and filename have been set.

10. Set **Connection** to **Texas Instruments Simulator** and check the **Device** that says **F283x CPU Cycle Accurate Simulator** as shown in the snapshot below. Select **Save** and then you can close the file (by selecting the X to the right of the NoBoard.ccxml below the toolbar). NoBoard.ccxml should appear on the left in the Project Explorer window, at the bottom.



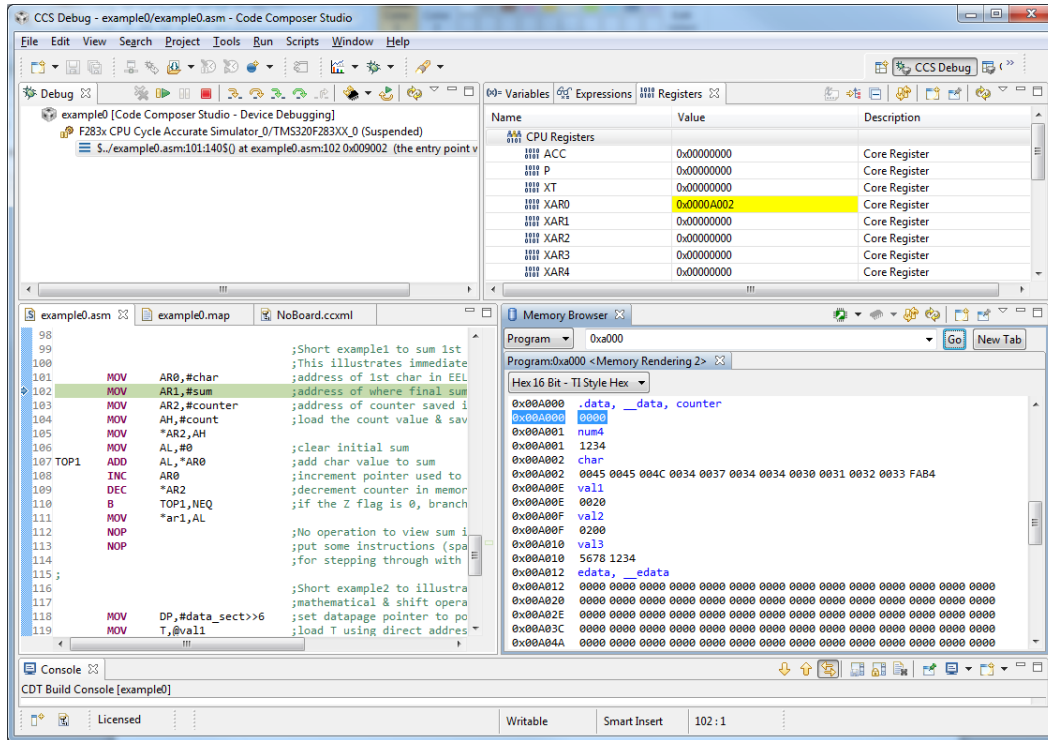
11. Finally we are ready to build (assemble & link) the project. Select the hammer icon , or **Project** → **Build Project**, or right click the project in the Project Explorer window and click **Build Project**. You should see zero errors and warnings when you build the project containing ex0.asm (see below) and you should also see that a machine code output file is generated called example0.out (See “Finished building target: example0.out” in the console window.)

## Creating and Simulate an ASM Project in CCS



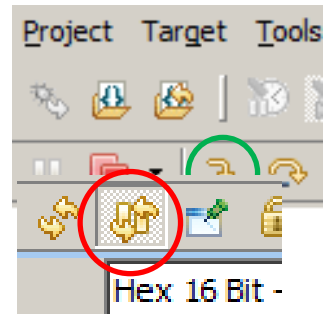
12. Finally we can start the **debugger** by pressing the small icon that looks like a six-legged insect shown to the right (or by selecting **Run** → **Debug** or with the F11 key). Do so now. A new window may pop up that says **Variables | Expressions | Registers**. If so, select **Registers**; if not, go to **View** to open a window for **Registers**. Go to **View** to open another window for **Memory Browser**. Expand the CPU registers icon to see the cpu registers. i.e., ACC, P, XT, ..., XAR7, PC, IC, etc. Type **0xa000** in the field on the **Memory Browser** window and set the pull-down next to this field to **program**. Select the downward facing arrow in the **Memory Browser** window and select **Default Rendering** → **Memory Rendering** (not **Traditional**). Then select **go**. You can see the variables that were placed into memory using the assembler directives in the data section of ex0.asm.





Scroll down to the **PC** (program counter) in the **Registers** window and verify that it is set to **0x9000**. This is where our code begins in ex0.asm. Begin stepping through the code by selecting the **step into** arrow (circled in green on the right in the **Debug** window) that is just to the right of the red **Terminate** square (used to exit debugging).

Scroll up to ACC, XAR0-XAR2 registers (in the **Registers** window) and observe how they change when the code is single stepped. Enter **0xb000** into the memory browser. You need to also select **Enable Continuous Refresh** (circled in red in the right snapshot) in the memory box (near the right edge) to see the memory changes when you step your code. Step through the first example that adds three ASCII characters. Notice that 0x00D6 (the sum of the three characters) is written into 0xb003 (which is the address associated with the sum label) after the **MOV \*AR1,AL** instruction is run.



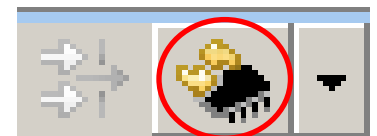
13. Single step through the remaining code and again observe the results being written into memory locations 0xb000-0xb003. *This program will be covered in great detail during our class lectures so it is very important to attend class to learn about the F335 addressing modes and instruction set.*

14. To add a breakpoint, select the line number in the asm file. You can view and delete the breakpoints by selecting **View → Breakpoints**. You can now run (the green arrow circled in red, as shown here) and it will stop at the breakpoint. You can either run again from there or start single stepping from this point forward.



15. To see the addresses for each of the instructions, **View → Disassembly**. This is very useful to confirm that everything is going where you want it to. To see the actual assembly language **including** the labels, select the icon in the disassembly window immediately to the right of the “Enter location here” box.

16. If you would like to run the program again, select the **Reset CPU** icon (shown to the right, circled in red) and then press the **Restart** icon



## Creating and Simulate an ASM Project in CCS

(immediately to the right that shows a green arrow with a yellow arrow underneath). Check to make sure the PC is starting again at 0x9000. A **Dissassembly** window may pop up which blocks the **Registers** window you opened earlier. You may close this but first notice that this shows the associated machine code for our program beginning at 0x9000. A **Console** window might also pop up; you can close this as well.

To terminate the debug session, press the red **Terminate** square. Note: Sometimes it takes a few seconds to exit debugging (due to CCS variables being removed from operation) so patiently wait until CCS returns to code assembly/link mode.

17. If you want to also debug **WITH** your board (i.e., emulate), you will need to make another Target Configuration. See *Emulate\_CCS\_Project.pdf* for details.

This ends the project creation and simulate tutorial. Refer to the emulate tutorial to run & test your code on the actual hardware. Also, see the Frequently Asked Questions section below for more information relating to saving time when performing CCS code development.

### FAQ Relating to ASM Project Creation & Build

1. Do I always have to save my ASM file every time I make a change before building?

*No, every time you press the Build Active Project icon, your ASM file is automatically saved first.*

2. I want to start writing & debugging & emulating code right away (on the actual board!). How do I start the emulator debugger?

*See *Emulate\_CCS\_Project.pdf* for more information.*

3. Do I need to create a new workspace for every lab?

*No, you can (and should) create all you projects or labs in the same workspace. You can just hide the projects you are not using by pressing the "-" symbol to the left of the project name.*

4. Do I need a different Linker Command File for every project?

*No, instead point the linker to the same file *KG\_RAM\_Link1.cmd* used in your first project. Or simply copy the *KG\_RAM\_Link1.cmd* file into each new project directory and point the linker to it.*

5. Can I create two or more projects in the same directory?

*No, you need to save each project in a different directory. If you try to save two projects in the same folder, an error message indicating that your new project overlaps the location of another project will be shown in the screen. It is recommended that you create a directory for each lab (i.e. labs 0-9) at the beginning of the semester so that when creating a new project, all you have to do is select the specific directory created for that project.*

6. When I have multiple projects open, how do I choose which project is active?

*This is performed by right clicking on the project that you want to make active and select **Set as active project**. All other projects will be ignored when you build the active one.*

7. When I re-run the debugger to execute my code, I noticed that the XAR0-XAR2 registers already had non-zero values in them. So I zeroed them out by clicking on them and then ex0.asm did not run properly. Why is this? What happened?

## Creating and Simulate an ASM Project in CCS

*When you re-run the debugger by pressing **Reset CPU** and **Restart**, several instructions are pre-fetched and executed due to the pipelining architecture of the CPU. Note: CPU pipelining will be further discussed in class. Instead you should reset & restart and simply leave the registers alone and step your code as before.*

8. When I step through my code, I noticed that it sometimes takes a couple more instructions to execute before results are actually written to memory. In other words, even though I stepped through an instruction that moves a register value to SRAM, I don't see memory change until a couple more instructions are stepped.

*Again I believe that this is due to the pipelining nature of the CPU. Most likely it was using the CPU address and data bus to pre-fetch new instructions and therefore waited several cycles later to perform the memory write cycle. This will effect when we view our results, however it is not a problem once we are running at full speed on our board.*

9. I am an engine-erd and I love this new tool. Someday I will create new products with this DSP so that others will have to work for me. Therefore I would like to know how to debug ex0.asm on my lab board so that I can begin experimenting with real hardware. How can I do this?

*Make sure the debugger is terminated by pressing the red **Terminate All** icon. Next plug in your lab board and right-click the **example0.ccxml** target configuration file and **Set as Active Target Configuration**. We are telling the CCS application that we now want to use our lab board as the target device for download/debug. Now press the debug (insect) icon as was performed earlier in simulated mode and step code as was done earlier in this tutorial.*