ECE 3710 Lab 1 – Fall 2017

Due Date: Week of September 18 at the beginning of your lab section (20 points)

Objectives

The student should become familiar with the process of building, downloading, debugging and running programs on a microcontroller using an IDE.

Overview

In this lab we will be running our first assembly program on the microcontroller and de-bugging another program that copies data in a couple ways. All assembly programs may be found on the course wiki.

Preparation

1. Come to lab with your Tiva C Launchpad board and its USB cable.

2. Be prepared to use most of the instructions you have learned in class.

3. Read Introduction to Arm 2.3-7 and Interfacing to ARM 1.3

4. Read through the two Keil uVision tutorials from Circuits Today posted on the wiki at https://spaces.usu.edu/display/ece3710/code_examples.

Procedure

1. Setup *Blinky*

    a. Download a blank TM4CGH6PM_ASM project from https://spaces.usu.edu/display/ece3710/Lab+Supplements

    b. Download the Lab1 code from the wiki and unzip the folder. There you will find ***blinky.asm***. Use it to replace the main.s in the TM4CGH6PM_ASM folder.

    c. Open the ***SP*** project.

d. On the left panel will be main.s under Source Group 1. Right click on main.s and remove it. Right click on Source Group 1, and add existing files to group. Add your ***blinky.asm*** file (you may have to change ***Files of type:*** drop menu to see .asm files).

e. From the menu, select **Project -> Build Target**. As long as there weren't any errors, your project is ready to run!

2. Running *Blinky* on Your Board

a. You will be using the onboard debugger to download code to and receive diagnostics from your board. The software drivers should already be installed on your host PC. If not, notify your TA.

b. Select **Debug -> Start/Stop Debug Session** and your debugger will automatically launch. In the debugger you have access to all of the memory on your device. You can step through the code using **F10/F11** or you can let it run by pressing **F5**.

c. Select **View -> System Viewer -> GPIO -> GPIOF**. This is the port which the LEDs on your physical board are hooked up to. In the GPIOF window, look at the DATA register. This is where your microcontroller will actually assert the pins which make the LEDs turn on and off. Pin F1 is mapped to the red LED on your board.

d. Press **F5** to run your program if you haven't already. Note the changing port pins. Step through the assembly code and see if you can understand what is happening. ***HINT: Section ten of the Tiva™ C Series TM4C123GH6PM Microcontroller Data Sheet on the wiki may prove helpful.***

e. As the last step in this section, use the logic analyzer to measure the frequency of the LED blinking, and modify ***blinky.asm*** so your board blinks twice as fast. Refer to the Tiva C Series TM4C123G LaunchPad Evaluation Board User's Guide on the course wiki to obtain pinout information for the red LED light. Take screen shots of your measurements and show the verification from the logic analyzer to your TA. ***NOTE: Make sure you understand how to connect your board to the logic analyzer. Ask your TA if you are unsure BEFORE connecting.***

f. For your report, include a screen capture of the timing before and after the code edits. Use the logic analyzer to obtain this information.

3. Commenting Code

   a. The *Blinky* code functions but is missing some important information. Fully describe the function by filling in the blanks in the comments. Clarity through commenting will simplify the debugging process throughout your career. Your freshly commented code will be part of the lab report.

4. Running *Blinky* Using the Simulator

   a. It isn't always practical to use/have a physical board for debugging purposes. Keil uVision is capable of simulating certain boards so that we can write and debug programs and then transfer them to a board later and know that they'll work. Unfortunately, the Tiva C isn't yet emulated in the Keil software, so we'll have to simulate a very similar board, the TI LM3S1968.

   b. Download the blank LM3S1968_ASM project from the **Code examples** page of the wiki (https://spaces.usu.edu/display/ece3710/code_examples).

   c. Use the ***blinky.asm*** file to populate the empty project.

   d. Build the project.

   e. Now select **Debug -> Start/Stop Debug Session** and your debugger will automatically launch for the simulated board. You can step through the program and examine registers/memory just as you would for a real board.

5. Code Debugging

   a. Now it's your turn to fix some code. Copy the file hello_students.asm from the wiki and set this up the same way you set up *Blinky* in an LM3S1968 project. This program copies a message "Hello Students!" from read-only memory to a location in SRAM above the memory reserved for the stack. However, there are bugs in the code which keep the program from functioning normally.

   b. Use the simulator to step through the code and figure out what's going on. For example, where is the message located in memory and where is it being written? The memory tab (lower left hand corner) gives a view of the device's memory. Once you find and correct the bugs, demonstrate to the TA that your program functions properly—i.e. you need to demonstrate that message is being written to memory correctly and explain what's going on.

6. Logic Analyzer

a. Using the logic analyzer to verify a bit stream is a very valuable skill. The objective of this exercise is to teach you to capture data by triggering on a bit pattern. Read “Trigger Setup” (p. 87—93) in the MDO3000 Series Mixed Domain Oscilloscopes User Manual (on the course wiki under Supplementary material for lab)

b. Download a blank TM4CGH6PM_C project from the course wiki (https://spaces.usu.edu/display/ece3710/code_examples). This is a C project, not ASM like before.

c. Replace ***main.c*** in the blank project with the ***logic_analyzer.c*** provided.

d. The logic analyzer file writes 8-bit parallel data on pins PE0 –PE3 and PC4-PC7, with PE0 as the least significant bit (LSB) and PC7 as the most significant bit (MSB). PA7 is the clock. Connect the logic analyzer to these pins, using port 8 to connect to the clock. **Again, verify your method with your TA before connecting if you are at all unsure.**

e. Modify Bus1 to match this configuration (Refer to Section “Setting Up a Serial or Parallel Bus” (p. 67) of the oscilloscope user manual to help you do this). The data is clocked, on the positive edge of the clock.

f. On the Trigger menu, define trigger type to be ***Bus***

g. Use the ***Data*** tab in the Trigger menu to begin storing data after receiving ASCII character \n (0x0A).

h. Run this configuration to capture the encoded message. The message will repeat infinitely, and ends with the \n character.

i. Include the bit stream in your report. Use the bit stream to decode the ASCII message.