August 29th, 2017

Lab 01 – Introduction HC11 Development

Zachary Davis

Christopher Bywaletz

**Introduction to HC11 Development**

**Introduction**

The purpose of this lab is to familiarize myself with the main components of the laboratory for the coming labs. There are three major components that were focused on in this lab. The develop board itself, BUFFALO, and the AsmIDE.

The first of which is the EVBplus2 68HC11 Board. This is a development board that provides easily modifiable program space, the ability to monitor the program, and to communicate with the desk PC. The board has many target applications like buttons, switches, and LEDs and all together can be used to test, debug, and modify applications before they are ready to be burned into EPROM. Within the EVB board is BUFFALO or, Bit User Fast Friendly Aid to Logical Operations. This is what allows us to monitor the board with nineteen commands that range from modification to the downloading of a program. Finally, the AsmIDE or, Assemble Integrated Design Environment. The is the development environment that allows us to write and compile source code for the HC11 as well as to communicate with the board through a serial connection.

As mentioned earlier the purpose of this lab is to gain a familiarity with the three major components of the lab. More specifically, a familiarity with AsmIDE so that we can comfortably create, write, and compile source code as well as download and communicate with the board. We also gained a familiarity with BUFFALO, which is what allows us to command the board like download or running a program. Most importantly the goal of this lab was to understand and familiarize ourselves with the EVB board, BUFFALO, and the AsmIDE and how all three work together in the development exercise.

**Procedures and Discussion**

The procedures in this lab were split into three sections: (1) Familiarizing Myself with the EVB Board and Testing Sample Programs, (2) Learning and Trying BUFFALO Commands, (3) The manipulation of memory on and EVB Board.

Up first is the familiarization with the EVB Board, learning where and how it is powered as well as its serial COM port. We were able to connect the board and establish communication through the serial communication ports by point the desk PC in the right direction in the settings of the AsmIDE. Once this far, we were ready to begin loading and modifying programs on to the EVB Board to experiment with its uses. To do this we would open four sample programs provided and run through the practices of compiling the program, preparing the board with the command LOAD T in BUFFALO, assembling the program, and finally using the G {Mem Location} command in BUFFALO to begin the program on the board. Doing this so many times over and over again with code we know to work was a great way to practice and learn the process. Finally, we manipulated switches, and observed as the EVB board ran the sample code and made note of what each program did.

The second portion of the lab focused on BUFFALO commands and CPU registers. We learned the RM command that allows us to see the values of these registers and how to modify them as to get the desired results if needed in future labs.

The final portion of this first lab focused on memory modification. First, using a memory dump command we learned how to view a certain block of memory rather than any modification. We then learned how to modify a specific bit of memory with the MM command. However, we also learned the commands limitations as well as further understanding the purpose of the different components of the board when try to write to read only memory. Finally, in this section on memory we learned how to fill an entire block of code with data rather than just small segments. The usefulness of this is excellent when for example you want to clear the RAM on the board for a new project.

Relatively this lab is much simpler and rather different than what is expected in the future, but that does not mean it is without problems. As a group and class, we were experimenting and learning with new never before used components and this means that many times I was unaware of what do to or even if I had done the task correctly what the results meant. Somewhat unique to this lab, my lab partner and I discussed with other classmates to solve the problem like what the memory dump represented.

**Conclusion**

This report has discussed the introduction of the HC11 to me and my lab partner. The goals of this lab were to gain a familiarity with the EVB board, BUFFALO, and AsmIDE and that is exactly what happened. In the following weeks lab, I will feel like I have a great enough understanding to begin using the board for is genuine purpose and can being development with the EVB board. That of course is not to say that I know all there is to know. This lab did not review much if anything about the development language used in AsmIDE, which will play a major role in the future. Also, knowledge is not understanding and knowing how to use a board definitively does not mean that I have an inclusive understanding of how to use it yet, but that will come with future labs.

**References**

Not Applicable

**Appendices**

Not Applicable – The only Hardware used was the EVB Board and the source code was the sample programs.

**Lab Questions**

Record Observations of Example programs.

EX1: Each of the eight switches corresponds with one of the eight LEDs on the board. By toggling the switch we are able to illuminate particular LEDs. Odd note about the program the switch state labeled “ON” is off and vice versa.

EX2: The LEDs count in binary (ON representing a 1 and OFF a 0) from zero to two-hundred fifty-five. This is done rather fast by default and we had to turn down the speed to see what was happening.

EX3: The on-board display of the EVB board displayed the word “help.”

EX4: The speaker, that until this point I did not know existed, played a short snippet of the son The Fur Elise.

What does the command RM do?

By typing out this command we can see the values/states of the seven CPU registers. With the command RM A we are even able to change the value of the A accumulator. This means that with RM “~” any of the CPU registers can be manually modified.

What do you think the data in this memory block represents?

**D000 8E 8F FF 86 00 BF 10 07 B6 10 03 B7 10 04 7E D0**

**D010 08 4F B9 4B 3F 47 06 43 09 37 46 00 00 00 00 00**

**D020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00**

The line and next couple segments represent the machine language code for the program EX1. For example, 8F represents lds and 8F FF represents #Stack and so on and so forth. However, there comes a point where I cannot match up the machine language with source code, but numbers are still present (nonzero). This is data from previously run programs that have not be re-written yet. (perhaps a strong argument for block fill in the future.)

What happens when trying to modify a ROM address?

This is of course restricted because of exactly what ROM is. Read only memory cannot be written do and the BUFFALO command too will not let us.

**Lab Participation**

This lab was relatively easy to split the work in half. The goal of the lab was to become familiar with all the common equipment in the lab so the only successful way to do that for both me and my lab partner is to continuously switch roles. That is exactly what we did. I would be manipulating the physical parts of the board for EX1.asm and he would be on the desk PC using the AsmIDE. Then for the next program we would switch and so on and so forth as to both have equal chance to learn and experience all the components. It is also worth mentioning during the memory dump portion of the lab multiple lab groups collaborated to try to answer the question that posed to us.