ECE 375: Computer Organization and Assembly Language Programming

Lab 1 – Introduction to AVR Development Tools

SECTION OVERVIEW

Complete the following objectives:

- Download and compile the sample AVR assembly source code given on the lab webpage (`BasicBumpBot.asm`).
- Understand how to connect and operate the AVR programmer.
- Upload the previously-compiled sample program to the flash memory of the TekBots AVR microcontroller board (`ATmega32U4`), and observe it running.

PRELAB

You will be required to complete a prelab assignment before attending the lab session itself. These prelabs will cover the important concepts and background knowledge necessary to accomplish each corresponding lab.

Prelab assignments are due before the beginning of your lab session each week. If you have not submitted your prelab by the starting time of the lab session, you will receive no credit for that prelab. For general information about assignment submission policies, please refer to the syllabus on the lab webpage.

For this first lab, there is no prelab assignment.

PROCEDURE

Compiling an AVR Assembly Program

1. Download the sample code (`BasicBumpBot.asm`) as well as the pre-compiler directive file (`m32U4def.inc`) available on the lab webpage. This is a simple AVR assembly program that is well-commented and ready to compile. All code that you produce for this course should be as well-commented as this code. Save this code somewhere you can find it.
2. You will be using `Windows`, `Linux`, or `Mac OS` system to develop your AVR assembly code throughout this course (with the exception of Lab 2, which uses C). You will be using it to write assembly programs for your AVR microcontroller board, which uses an ATmega32U4 microcontroller.
3. Now that you have loaded the `BasicBumpBot.asm` and `m32U4def.inc` into the same directory, take a moment to look over the sample code. Although you haven't seen any AVR assembly code in this class yet, read the comments and see if you can follow the general structure and flow of the program. For reference, the general operation of the program is described in Figure 1.
4. By following the intallation guide provided on the lab webpage, learn how an AVR assembly program is compiled. The resulting output is a binary program file (called a HEX file, with a `.hex` extension). This HEX file contains the actual binary instructions that will be executed by the ATmega32 microcontroller.

- Initializes key components of the ATmega32
- Starts the TekBot moving forward
- Polls the whiskers for input
- If right whisker is hit
  - Backs up for a second
  - Turns left for a second
  - Continues Forward
- If left whisker is hit
  - Backs up for a second
  - Turns right for a second
  - Continues Forward

Figure 1: Theory of Operation for Lab 1 AVR Assembly Code

Uploading a Compiled AVR Program

1. With the HEX file uploaded to the microcontroller, you can observe the behavior of the sample program. The LEDs on the `mega32` board should

indicate that the board is performing a basic BumpBot routine (remember, Figure 1 gives a description of the full routine). To receive your implementation points for this lab, modify the sample program so the TekBot will reverse for **twice as long** before turning away and resuming forward motion. Demonstrate to your TA that your board is performing this correct BumpBot behavior.

STUDY QUESTIONS / REPORT

For every lab, you will be required to submit a write-up that details what you did and why. As a requirement of Lab 1, you need to fill the program overview section with a summary of what you have understood by reading comments in the example AVR code. A detailed description of each routine is also required to be elaborated. You also need to answer the study questions given later in this document. Provide your answers using the "Lab Report Template" provided in Section 5 on the lab webpage. **You must use the template**. You must submit your write-up report and the AVR assembly code via **Canvas** before the start of the following lab. **NO LATE WORK IS ACCEPTED**.

Code that is not well-documented will result in a severe loss of points for the write-up portion of your lab grade. For an example of the style and detail expected of your comments, look at the code you just downloaded. Generally you should have a comment for every line of code.

Study Questions

Most of the labs you do in this class will have study questions that are to be answered within your lab write-up. This lab's study questions are given below, and will be due at the start of lab next week. As is often the case in engineering courses, there will be some occasions when you are exposed to information that has not been covered in class. In these situations you will need to get into the habit of being proactive and using your study skills to research the answers. This can involve strategies such as reading ahead in the textbook, checking the datasheet, searching online, or reviewing other documentation from the manufacturer.

1. Take a look at the code you downloaded for today's lab. Notice the lines that begin with `.def` and `.equ` followed by some type of expression. These are known as **pre-compiler directives**. Define pre-compiler directive. What is the difference between the `.def` and `.equ` directives? (HINT: see Section 5.4 of the AVR Assembler User Guide).

2. Take another look at the code you downloaded for today's lab. Read the comment that describes the macro definitions. From that explanation, determine the 8-bit binary value that each of the following expressions evaluates to. Note: the numbers below are decimal values. The answer must be written in binary.

   (a) (2 << 6)
   (b) (3 << 3)
   (c) (8 >> 1)
   (d) (255 << 0 & 15 << 4)
   (e) (8 >> 3 | 7 << 4)

3. Go to the lab webpage and read the AVR Instruction Set Manual. Based on this manual, describe the instructions listed below.

   ADIW, BCLR, BRCC, BRGE, COM, EOR, LSL, LSR, NEG, OR, ORI, ROL, ROR, SBC, SBIW, and SUB.