EE 333 Microcontroller Engineering
Oregon Tech Portland, Fall 2013
Lab Assignment #1 — Familiarization
Due October 17

Objective: This student will become familiar with the basic tools used in microcontroller application — AsmIDE programming environment, AS12 assembler, D-Bug12 debugger, 68HCS12 simulator, and the Dragon12 experimentation board.

Equipment and Software needed:

- Dragon12-Plus2 board and USB cable. You don't need to use the separate power supply.
- Computer running Windows, or classroom computer
- USB Flash drive if you use the classroom computer
- D-Bug12 command documentation, provided in the course distribution
- Simulator and AsmIDE installed on computer

General Instructions: You should create separate folders on your hard drive for each lab assignment. If you use the classroom computer, copy all the source files to a USB flash drive at the end of the session. **Do not rely on the files being there the next time you are in class!** Some students have used a USB flash drive as their working drive in class. This is not recommended because it is much slower and puts more wear on the flash drive. It is better to use the hard drive and copy the files to the flash drive when finished.

Requirements of the lab report (your submission) vary between assignments. This first assignment requires a report that is much longer than those that follow. The goal here is to make sure you can use the tools (the 68HCS12 simulator, AsmIDE, and the Dragon12 board) for developing the projects in the remaining lab assignments.

Please keep your report "readable." I've got a lot to read in little time (I always grade quickly) and a sloppy report is discouraging. In your lab report you should use the Courier font (or some other mono-spaced font) for program listings and transcripts, and some other font like Helvetica, Times Roman, or Microsoft's Calibri or Cambria for body text.

This is Courier New. Each character takes the same amount of space. This is Helvetica. This is Calibri. This is Times Roman. This is Cambria. Use one of these for body text, or something similar. I use Museo, which I've purchased, so my assignments look distinctive yet still very readable (I hope!) This is Comic Sans. Don't use it in your reports!

There are videos on Blackboard that have an example of usage of AsmIDE, the simulator, Dragon12 board, and debugger that should be viewed prior to starting this assignment. It will also be demonstrated the first class session. You should be able to start working on this assignment after the first class session. Please don't wait until the last minute, because if you get stumped by any part of the lab you will have the second class session to ask questions and get your problems resolved. The difficulty with this assignment lies solely with getting all the tools operational and understanding what they do.

The first part of the lab uses the simulator, so you can get started even if you do not have your Dragon12 board yet. If you haven't ordered the board, order it now from http://www.evbplus.com. Make sure you buy the version that has D-Bug12, which is the Dragon12-Plus2-DB. Don't buy the Dragon12-Plus2-SM.

In the 68HCS12 microcontroller used on the Dragon12 board and simulated by the simulator, the I/O registers start at location 0 (do not attempt to put a program there!), RAM memory starts at $1000 and can be used through $3BFF. The remainder is reserved by the debugger, D-Bug12. As a rule, we will store data starting at $1000 and the program starting at location $2000. While "in real life" a program would be in EEPROM or Flash ROM memory, we won't do that here until the end of the second term. At the moment, the Flash ROM contains the D-Bug12 program and that program is easiest to use when your program is in RAM.

PART 1 — Using the Assembler and Simulator

In this part of the assignment you will use the assembler to convert the program from assembly language into machine code. then you will test the program using the simulator.

Using the editor in AsmIDE, create a new file and save it as *lab1.asm*. If AsmIDE asks about setting up your serial port, cancel out as you can't do that at this time. By first saving the file you will get color coding in your source file. Enter the following assembler source code. Note carefully that the *loop:* label must start in the first column while all other statements must NOT start in the first column. Everything after the semicolons is a comment. You don't need to copy the comments into lab1.asm if you don't want to, but all programs you write need to be commented.

```
      org   $2000 ; load program starting at location $2000
      ldaa  #$20  ; load accumulator A with the value $20
loop: inca        ; increment the value in A
      staa  $1000 ;   and store into location $1000
      bra   loop  ; branch back to "loop"
      end
```

It is important in the future that all assembler source files you create have names ending with ".asm" because any other ending will prevent the assembler from working properly. Also file names must not have embedded space characters. Use

the underscore character to separate parts of a file name. For example, name the file “lab2_part1.asm” instead of “lab2 part1.asm” or “lab2_part1”.

Assemble the program. This will produce two new files, *lab1.s19* which contains the machine code in a format that can be loaded into the Dragon12 or the simulator, and *lab1.lst* which is the assembler output listing that is “human readable.” Look at this file and see for yourself — it is the source file but has additional columns which give the addresses where the instructions are stored as well as the bytes that make up the instructions. If any errors are listed in AsmIDE, correct them now. You cannot proceed if there are any errors.

Start the simulator program in *student* mode. This mode simplifies operation and will be useful for this lab assignment as well as the second and third homework assignments. From the File menu, load *lab1.s19*. Step through the program to verify operation. You should familiarize yourself with the following controls at this point:

- Step
- Go
- Stop
- Display Updates
- Source
- the Memory Display fields (you can click in the data area to change the values, by the way).
- Also look at the Code Inspector. If you click on an instruction in the Code Inspector it turns on or off a breakpoint at that instruction. Execution stops automatically when a breakpoint is reached.

You can use Tools menu — Make Snapshot and View menu — Console Log to save simulation information to the clipboard, where you can paste it into your lab report. Describe what you did in your lab report.

PART 2 — Verification of the Dragon12-Plus Board

In this part of the assignment AsmIDE will be used as a terminal emulator. You will need to install the driver for the USB to serial port adapter which is built into the Dragon12 board. There is a file in the Dragon12 documentation with the installation instructions. With the USB cable connected, start AsmIDE. You will need to go under the Options menu to configure the serial port number, the communication rate to 9600 bps, 8 bits data, no parity, and Xon/Xoff flow control. Open the terminal pane (there is a button on the button bar to do this).

Connect the power supply and verify the LCD says “Dbug-12 EVB MODE.” This should be the case if the switches on SW7 are both in the down position.

If this is not displayed, there is a possibility that the board was ordered for Serial Monitor rather than D-Bug12. In that case the flash memory will have to be

reloaded. I can do that in the classroom. You can also do this yourself following the instructions provided with the Dragon12-Plus.

If all goes well, hitting the Enter key on the keyboard should cause the Dragon12 to respond with the D-Bug12 prompt, ">".

Become familiar with the following D-Bug12 commands by executing them and verifying that they work. **In your lab report, explain how you use these commands and show the results of using each of them.** You will be using these commands throughout the term and in EE335, so taking the time now to understand what they do will save you far more time later on when you are debugging and analyzing programs.

- HELP — List commands
- BF — Block fill memory with data
- MD — Memory Display
- MM — Memory Modify
- MOVE — Move a block of memory
- RD — Register Display
- RM — Register Modify
- <register name> <register value> — Quick way to change a register's contents

Note that for all of these commands, values are in hexadecimal. **Do not** use a leading dollar sign when you enter values to D-Bug12.

As an example of showing the results of using a command, to show how the BF command works you could use the MD command to show a region in RAM, then execute the BF command to modify a portion of the region, then use the MD command again to show that only those locations changed. You can copy the terminal window into a Word document to save the transcript in your report.

PART 3 — Running a Program on the Dragon12-Plus

In this part of the assignment you will enter a program directly into the Dragon12 RAM and then execute it using the D-Bug12 monitor program. You will use the following D-Bug12 commands — ASM, G, T, BR, and NOBR.

Connect the Dragon12-Plus to the computer and start AsmIDE as you did in Part 2.

Read the description of the D-Bug12 ASM command and then use it to enter the following program starting at location $2000:

```
INY
DEX
ADDA #$10
BRA  $2000
```

You exit the ASM command by entering the command "." (just a period). Note that within the ASM command literal values are in decimal and you need the $ to indicate values in hexadecimal.

Examine the registers then use the G command to start execution at location $2000. Stop the program using the ABORT button on the Dragon12-Plus. What do you observe about the register contents?

Trace the program execution by using the T command repeatedly. What do you observe?

Use the T command to advance to location $2000, then use the BR command to set a breakpoint at the ADDA instruction at location $2002. Resume execution with the G command. What happens? Execute the G command again. What happens?

Remove the breakpoint with the NOBR command then resume execution. What happens?

Note that a bug in D-Bug12 will cause breakpoints to be ignored if instruction after the current one to be executed after the G command has a breakpoint set.

Observe the difference between stopping execution with the RESET button instead of the ABORT button. What is the difference?

One final note — every feature you have used in D-Bug12 (except for the ASM command) is also available in the simulator, and typically is easier to use. So if your program doesn't work, the simulator is often the best place to see what it is actually doing. However, remember that a simulator doesn't simulate everything on the Dragon12 board, nor is it reality. You should always finish a lab assignment by running your program on the Dragon12 board.

To turn in

Your lab report, as a single PDF, DOC, or ODT file. The report should have in it:

- Screen shots or "snapshots" of your use of the simulator, with an explanation of what you did.
- Transcripts of your use of D-Bug12, with an explanation of what you did to check out each command.
- Answers to all questions posed in the assignment.

Note that future assignments do not require transcripts or screen shots unless specifically requested!