Lab 5: Introduction to Programming Microcontrollers

Summary:

Demonstrate the process of writing, assembling, downloading, and monitor-level debugging of assembly language programs using the Axiom Manufacturing CML12S-DP256 development board for the Freescale 68HCS12 microcontroller. Use the development board hardware to demonstrate the operation of practical program functions including delay loops and data transfer to microcontroller I/O ports.

Learning Objectives:
- Review of assembly language syntax
- Introduce to the MON12 monitor program
- Introduce to the CML12S-DP256 development board
- Gain experience with assembly programming and delay loops
- Experience hardware and software protocols of standard port I/O

Resources and Supplies:
- Text editor & HC12/S12 assembler (WinIDE development environment)
- MON12 monitor program
- CML12S-DP256 development board
- *CML12S-DP256 MON12 manual*

All documents are available on the class website

Important Reminders:
- It is your responsibility to save the programs you create.
- Pre-lab assignments must be completed <u>before</u> coming to the lab.

****Lab Teams****
- This and all subsequent labs will be completed in teams of two. You may choose your own lab partner. See the TA if there is any confusion with team assignments. Students may work alone if they desire but are still responsible for completing the lab within the designated lab period.
- Each **individual student** should complete his/her own **pre-lab**, lab **check-off sheet** and **lab report**.

Background:

CML12S-DP256 Development Board and User Interface Hardware Overview

Axiom's CML12S-DP256 is a development board for the Freescale MC6S12DP256 microcontroller which is a configuration of the 68HCS12 core with memory and I/O. The CML12S-DP256 and associated software, particularly the MON12 embedded firmware monitor program, provide a user interface to the microcontroller useful to debugging code and testing microcontroller hardware functions. Assembled code in .S19 files can be loaded to the controller through the MON12 program. Additional details can be found as needed in the *CML12S-DP256 MON12 manual.*

The CML12S-DP256 provides a debug mode in which the debugger software (e.g., MON12) handles CPU initialization, interrupt vectors, and the stack. The memory map for the CML12S-DP256 is shown on page 11 of the manual. The manual suggest using external RAM starting at $4000 for storing user program code in the debug mode of operation. It is suggested that program data be stored in internal RAM between $1000 and $3DFF The MON12 debugger reserves the remaining part of internal RAM between $3E00 and $3FFD and starts the stack at $3F80.

The ECE331 lab HCS12 microcontroller user interface hardware consists of two main components, a CML12S-DP256 development board and an AXM-0295 project development board. The CML12S-DP256 development board contains a 9S12DP256B microcontroller, which has a HC12 core and peripheral memory and I/O devices. The AXM-0295 project development board is multi-function user interface board with elements such as a liquid crystal display (LCD), LEDs, seven segment display, 40bit DIP toggle switch, 4x4 keyboard pushbutton matrix, etc. The two main boards are joined together by a couple of ribbon cables that connect I/O signals from the project board to/from microcontroller I/O signals on the development board. Using the user interface elements on the project board requires knowing what microcontroller ports they are connected to so software can be written to interact with these interface elements. It would be overwhelming to cover all of these connects at once, so they will be introduced as needed for each lab assignment. In this lab, you will interface with the LEDs. The information necessary to access these devices with microcontroller software is given below.

The project and development boards have a number of ‘headers’, connectors where wires can be inserted to access signals or where jumpers can be inserted to connect signals. For this lab, all of the signals that need to be connected between the project board and development board are contained within the ribbon cable between the boards. Thus, you will be able to access all hardware features of the project board through the microcontroller.

Loop Delays

From class you know how to calculate the time required to execute assembly instructions, including loops and nested loops. The table of HC12 assembly instructions lists the clock cycles require for each instruction. To calculate the delay in units of time, you also need to know that the HCS12 of the CML12S-DP256 development board operates at a ***clock frequency of 8 MHz***. However, the rate of instruction execution frequency may be only a fraction of this (like ½, ¼, etc.) so your calculations may not match exactly with hardware results. You will need to test your delay in lab and adjust your delay code.

Microcontroller Registers

The microcontroller contains many registers to control and interface with I/O devices such as data ports. For the MC9S12DP256B, a memory-mapped microcontroller, these registers are mapped to memory addresses $0000 - $02FF. Appendix C of the 331 textbook lists each of these registers. Rather than try to explain 100’s of register functions, we will introduce a few in this lab to give you a feel of their usage. Primarily, we will be using I/O port registers, which hold values written to/from the port and enable software to access (read or write) port data. We will also use registers that control the ports, for example, enabling their functionality or setting their data direction (input or output). The registers used in this lab are explained below.

Port K (LEDs)
The AXM-0295 project board includes 4 LEDs useful for visual display of program outputs. The LEDs are wired to the lower nibble (4 bits) of the 8-bit Port K of the microcontroller, as shown in Figure 1. The HC12 has two registers associated with Port K, a *data* register where values can be read from or written to, and a *data direction* control register that determines if each bit of the port is an input or and output. When a bit of the *data direction* register is set high (1), the corresponding bit of the *data* register is an output. If it is set low (0), the *data* register bit is an input. Thus, to set Port K to output signals to the LEDs connected to its lower nibble bits, a value of $XF would need to be stored to the Port K *data direction* register (where X means don't care for the upper 4 bits).

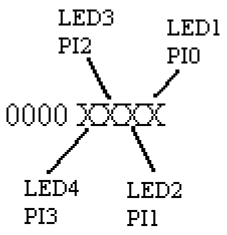

Figure 1. Port K data register.

The Port K registers are located at the following memory addresses:

$0032	Port K data register
$0033	Port K data direction register

These registers can be read or written to (using load or store instructions) just like any other memory address, but writing to the data register only affects bits set as outputs.

Pre-lab Assignment:

- Read this entire lab assignment so you know what to expect in the lab. Also flip through the *CML12S-DP256 MON12 manual* to become familiar with where to find information in that document. You do not need to learn the entire manual. The lab will walk you through what you need, but do look at the command list on page 10.
- Complete the steps described in the Pre-lab sheet near the end of this document. Each student must complete his/her own pre-lab before coming to the lab and hand it in to the lab TA at the beginning of the lab.

Laboratory Assignment:

- Print the check off sheet. Where indicated, you must record your results on the check-off sheet. After you successfully finish each part of the lab, show the TA your results and ask him to sign the check-off sheet.

Part 1: Loading and debugging ASM programs on microcontroller hardware

<u>Preparation:</u>

1. Create (or locate if you've already created one) a folder to store all of your ECE331 assembly programs. It is suggested you save these in your network drive space or

otherwise ensure it is stored off of the lab PC's hard drive. You are responsible for saving all of your programs, and they may get erased from the lab computers.

2. Open the ***WinIDE*** assembly programming environment and open the *lab5.p1.asm* file created in the pre-lab. This program should already be free of errors and tested for functionality. Assemble/Compile the to generate a .S19 record.

Note: The lab computers also have a programming environment called ***AsmIDE*** that will compile (by using the *Build > Assemble* menu command) code written in a text editor. Because WinIDE has a built-in editor and simulator, it is suggested you use WinIDE instead.

3. From the PC on your lab bench, launch the ***AxIDE*** utility from the 68HC12 program folder under the 'Start' menu.
4. Connect the CML12S-DP256 board to your computer using the wiring bundle attached to your computer.
5. Connect power to the CML12S-DP256 board. A window will pop up on the computer asking which communication port to use. Select **COM1**.
6. Press the **reset button** on the CML12S-DP256 board. In your terminal window on the PC you should see a message to "PRESS KEY TO START MONITOR…". Press the ENTER key and you should see:

```
Axiom MON12 - HC12 Monitor / Debugger V256.5
Type "Help" for commands.
>_
```

Note: Refer to page 10 of the *CML12S-DP256 MON12 manual* for additional information on any MON12 command or type 'help' at the prompt for a list of commands. The most useful command are:

MD <address>	***Memory display***
MM <address>	***Memory modify***
RD	***CPU Register display***
RM	***CPU Register modify***
G	***Go = run program***
T	***Trace = step***
BR <address>	***Set Breakpoint***

7. Type '**load**' at the prompt and press the ENTER key.
8. Select the '**Upload**' option. Then click on the '**Browse**' button and select the ***.s19*** file that was created from the assembly process in step 2. Once you have selected the file, click 'OK'.

Inspection & Testing:

9. Verify that the program is stored correctly in the memory using the MD command to display the contents of memory. ***MD <start_address> <end_address>*** will display the contents of the given memory address. Using the information in the CML12S-DP256 manual, see if you can display the entire block of memory that holds your program with a single command. Record the contents of memory addresses $4000-$4006 on the check off sheet. You may ignore data beyond $4006.

10. Comment on the check-off sheet how you can use these values to verify your program is correctly stored on the microcontroller.

Experiment:

11. Use the 'Modify CPU Register Contents' command (***RM***) to set the Program Counter to point at address $4000.

    > RM P <enter>
    > 4000 <enter>

Note: Use the **RD** command at any time to display the contents of the CPU registers.

12. Looking at the program and/or the .lst file, determine what memory address the final result is stored in. Record this address on your check-off sheet.
13. Now that the program and data are loaded and the CPU is ready to run your program, use the *trace* command (***T***) to step through your program one instruction at a time. Type **T** and press ENTER. This command should also update register values after each instruction is executed. Continuing to press ENTER should step through execution of each program instruction. After each instruction is executed, verify that the expected result was obtained, i.e. that CPU registers and memory contents are updated as you expect for each instruction. If there are any discrepancies, check the program contents at $4000 again to ensure you entered the program correctly and fix any mistakes. If you have to start over, be sure to reset the PC to $4000.
14. When the PC reaches $400E (SWI), the last instruction has been executed and the calculated result has been stored in memory. Use the **MD** command to display your result and record the result value on the check-off sheet.

Now that the program has been verified to work correctly, it is not necessary to step through it one instruction at a time. To run any portion of the program, you can set a *breakpoint*, an address where program execution should stop so you can observe the results. In general, you can set breakpoints anywhere in a program where you would like to stop execution and observe results.

15. Use the 'Set/Display user breakpoints' command (***BR***) to set a breakpoint at address $400E.
16. Looking at the program code, you should be able to see that modifying the data value at $6000 would modify the value that is repeatedly added by the program. Use the MON12 'Memory Modify Bytes' (MM) command to change the value added (num1) to **$10**. Type ***MM <address>*** *ENTER*, where address is the starting address of memory you want to modify. Then enter the new value at the prompt and hit ENTER after each value. To end entering values press the '**.**' key. Do not change the data value by altering the .ASM program and reloading.
17. After modifying the data memory, run the program again. Set the PC value to $4000 (**RM** command). Use the 'Begin/Continue execution of user code' command (***G***), and press ENTER to start the program execution. The program will run until it reaches the software interrupt instruction (SWI) near the end of your program or a user-set *breakpoint*. If you prefer, you can step through one instruction at a time using the **T** (trace) command.

18. Record the result value and address. Confirm this is the correct value before continuing.
19. Next, use the **MM** command to change the number of times the program loops (i.e., the count) to any number you would like. Notice that even though the count value is stored within the program (rather than in data memory), it can still be changed with the **MM** command during the debugging phase.
20. Record the memory address you changed, the value you changed it to, and the expected result value.
21. Run the program (do not forget to reset the PC value) and record the observed result value and address.
22. Ask the TA to check a demonstration of your program and check off Part 1 on your lab check-off sheet.

Part 2: Output Ports and Delay Loops

For Part 2, you must write a program that will sequentially count from $0 to $F and then loop back and restart from $0 again. The binary count value (0000 to 1111) must be displayed on the four LEDs of the project board. The program must provide a time delay of approximately 1 second between each count so that you can visually observe the LEDs at each count value. The program should begin at $4000. The Background section contains information you will need to complete this task.

Preparation:

1. Using the code blocks and pseudocode (or flowchart) you created in parts 2-4 of the pre-lab, write an ASM program to implement the LED counter described above. Although lab partners may have slightly different plans from pre-lab, work with your lab partner and quickly decide on a plan you both think will work well.
2. Use the WinIDE environment to write the code, compile it, and correct any syntax errors. Save the program on your network drive or somewhere else besides the lab PC hard drive.
3. Use the WinIDE simulator to verify your program is working correctly. Once you are confident your program is working, print a copy of the .LST file for each lab partner to include in his or her lab report.
4. The CML12S-DP256 board should already be connected to your PC and to power. If not, connect it now.
5. Press the **reset button** on the CML12S-DP256 board. In your terminal window on the PC you should see a message to “PRESS KEY TO START MONITOR…”. Press the ENTER key and you should see:

```
Axiom MON12 - HC12 Monitor / Debugger V256.5
Type "Help" for commands.
>_
```

6. To load your program, type ‘**load**’ at the command prompt and press the ENTER key. Select the ‘**Upload**’ option. Then click on the ‘**Browse**’ button and select the

.s19 file that was created from the assembly process. Once you have selected the file, click ‘OK’.

7. Use the ***MD <address>*** command to display your program code (starting at $4000) and verify your program is properly loaded.

Experiment:

8. Use the **RM** command to set the Program Counter to point at address $4000.
9. Type ***G***, and press Enter to start the program execution.
10. If your program is correct and your delay is sufficiently long, you should now see the count being displayed on the LEDs. If not, debug your program and repeat. Ask the TA for assistance if you are unable to fix problems yourself.
11. Once your program is correctly counting, record the count sequence in the check off sheet. If necessary, you can increase your delay loop delay time or start the program over if you ever lose track of the sequence.
12. When you are satisfied that the program is operating correctly, ask the TA to check a demonstration of your program. Ask the TA to check off Part 3 on your lab check-off sheet.

Final Tasks and Notes

- Turn off the power supply and anything else that you might have turned on.
- Disconnect and return the microcontroller interface board to the lab closet.
- Each individual student should prepare his/her own lab report.

Discussion Points

As explained in the *Lab Report Guide*, you should address these discussion points in a designated section of your report.

1. The program you prepared for Part 1 of this lab (pre-lab #1) contains the line
   ```
   count   equ     $03
   ```
   What would be the effect of changing this line to
   ```
   count   fcb     $03
   ```
   Describe in specific terms. Would any other lines in the program need to be changed to accommodate this new line of code?
2. In this lab you experienced both downloading programs/data to the microcontroller and directly modifying memory using MON12 commands. It is actually possible to enter programs directly into memory, byte by byte, using MON12 commands. Describe at least two advantages to downloading code rather than entering it byte by byte.
3. You now have experience writing simple programs that can make decisions, implement looping constructs, and write to microcontroller ports, which could be connected to various hardware devices such as the LEDs in this lab. Consider these experiences and think of another experiment you could perform using the tools and skills of this lab. Briefly describe your idea(s).

PRE-LAB 5

Due: At the beginning of lab.

Student Name: ___________________________ **Lab. Section (time):** __________

Please read the Background section and scan the lab assignment before you start the pre-lab. Although you will do this lab in teams of two, each individual student should complete his/her own pre-lab.

1. Prepare ASM Code for Part 1 of Lab 5
    a. Type the code below into an ASM editor/simulator. You can use WinIDE from the textbook CD, which is available in DECS labs and has been used in the PC lab lectures, or one of the freeware tools posted on the class website. Name your program ***lab5.p1.asm***.

```
;program code for Lab 5, part 1
        org     $4000
        ldab    #count
        ldaa    num1
more    adda    num1
        staa    result
        decb
        bne     more
        swi
        org     $6000
num1    fcb     $04
count   equ     $03
result  rmb     $01
        end
```

    b. Describe below what the program should do. Specify what result is expected at the end of the program and where (specifically) this result will be stored.

    c. Compile and simulate the program until it is free of errors. Describe below what you checked to ensure the program was functioning correctly.

    d. Save your .asm file so you can access it during the lab.
    e. Print the final .LST file and turn it in with your pre-lab.

2. Write the ASM code for delay loop with a delay of approximately one second. Determine the total number of loops necessary to create this delay explain how you computed this value.

3. Write all of the code (including port setup) necessary for sending a value of #$5 as output through Port K to display as a binary value on the project board LEDs.

4. Plan the full program needed for Part 2 of this lab by creating either a flowchart or pseudocode for the assigned tasks. Show this to the TA and ask him to initial it before beginning your lab. Attach the flowchart/pseudocode to your lab report. Note, you can write the ASM code if you want to, but you must turn in a plan (flowchart/pseudocode) and not just the ASM code.

REMINDER: Pre-labs should be completed by individual students and must not be copied between lab partners or friends. **Copying a pre-lab is considering cheating!** It will also degrade your learning experience in the lab if you have not done the pre-lab.

LAB 5 CHECK-OFF SHEET

Student Name: ____________________________ **Lab. Section (time):** __________

Complete this sheet as you complete the lab. Remember to have the TA check off each section of the assignment. This sheet must be included in your lab report.

Part 1: Loading and debugging ASM programs on microcontroller hardware

Step 9. The contents of memory in addresses

$4000______	$4003______
$4001______	$4004______
$4002______	$4005______

Step 10. Comment on verifying program is correctly loaded.

____

Step 12. The address where the result of the program is stored. ____________

Step 14. The result value of the program. ___________

Step 18. Result with num1 = $10: <address> _________, value = _________

Step 20. Modify count: <address> ________, value = _____, expected result = ______

Step 21. Modify count: result <address> _________, value = _________

Part 1: TA sign off
Part 1: Initial________

Part 2: Output Ports and Delay Loops

Step 11. What is the observed binary sequence on the LEDs?

_ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _

_ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _   _ _ _ _

Is the observed delay sufficient to display the count sequence? ________

Part 2: TA sign off
Part 2: Output ports and delay loops Initial________