# ECE9047: Laboratory 2

John McLeod

Due Date: 2024 03 14

## *Objective*

This lab is about programming an ARM®Cortex-A9 on a DE1 SoC board. For this lab, you can use an online simulator to test your code.

- The simulator is available at https://cpulator.01xz.net/?sys=arm-de1soc.
  - This simulator runs in your browser and should work with any computer system.
  - This simulator is *not* recommended for tablets, but might still work.
  - This simulator is reported to run best in Firefox. If the emulator doesn't seem to work well in your browser of choice (Safari, Edge, Chrome, etc.), try Firefox first before complaining to me.
- Alternatively, if you have (or are willing to purchase) an actual ARM®Cortex-A9 and a suitable development board, then by all means do so! Please take some pictures to include with your report.

The goal of this lab is to implement the following system:

> **Create a three-digit decimal counter that accurately counts 1 s intervals. The count should be displayed as a decimal number on the 7-segment display. An interrupt-enabled push button should be used to reverse the direction of the counter. The count should roll over to zero after reaching 999 (if counting up), or vice versa if counting down.**

Your program should run in a continuous loop.
You **must** use interrupts to handle the push button input. You may continue to use interrupts with the timer to handle the 1 s counter, or poll the timer for the 1 s interval. You **must** use the timer to count for 1 s.

## *Deliverables*

You must submit your lab report online to the course website on OWL. Your lab report must include:

1. A statement of the problem. (You can copy this from the *objective* given above.)
2. An explanation of your solution, outlining how you implemented the solution in your code.
   - A flowchart and/or a description using pseudocode may be appropriate, especially if your actual code involves lots of branches and subroutines, and/or lots of hard-coded numbers.

   The goal here is to make sure someone else (the TA, in particular) can understand how your code solves the problem.
3. A description of how you approached and implemented the problem: what did you implement first, how and when did you test whether your code worked, etc.
   - There are at least three separate parts to this project (pausing for 1 s, writing to the seven segment display, and accepting interrupts from the push buttons), unless you are very skilled at assembly you probably didn't write all of this code at once.
   - Reread my lesson notes from section 2.6 of Valvano's textbook on debugging theory (my notes are available on OWL) if you aren't sure what to do or why I am asking you to do this.

The goal here is to make sure someone else understands how, starting from a blank page, you ended up with some working code. You do not need to describe every little step and mistake that was made, just enough to give an overview of your design process.

4. A qualitative cost/benefit analysis of your solution, to help justify your approach. This discussion should be in terms of time spent designing your solution, time spent actually implementing this solution in code, and time that might be spent in the future maintaining or extending this code. The goal here is to make sure someone else understands your decision making process that lead to the solution you are providing.

5. A working copy of your assembly code. The TA is going to copy+paste this into the simulator to test that it works.

    - To make this as simple as possible, *please submit your code as a separate plain-text file (i.e. in TXT format, from Notepad or something equivalent)*.
    - Alternatively, if in-person learning has resumed, you may bring your laptop to class and demonstrate the working program to the TA or course instructor.

Your report should be in a common document format (probably PDF, DOC, or DOCX) and should be reasonably professional: written in complete sentences, appropriate spelling and grammar, and appropriate use of graphics. "Appropriate use of graphics" is deliberately vague and subjective.

- It is possible to produce an excellent lab report with no graphics at all, but often block diagrams, flow charts, and/or screenshots of the simulator may help clarify your written text.
- You should avoid placing full screenshots of the simulator if you are only calling attention to particular parts (i.e. sample seven segment display output, or sample register content, etc.). In that case, make sure you crop down the screenshot so only the relevant portions are shown in your report.

Your report should not be excessively long. Probably between two and five pages is a good guideline.

## *Grading Scheme*

Your report will be graded by the following scheme.

*5%* Statement of the problem. Marks awarded based on how clearly the problem is stated.

*15%* Explanation of your solution. Marks awarded based on how clearly you explain how your approach actually solves the problem, and on how easily the TA can tell that the explanation here matches the source code provided.

*15%* Description of your approach and how you implemented that solution. Marks awarded based on how clearly you explain, step-by-step, how you ended up with working code. Marks will be deducted if this section is too detailed (i.e. don't explain how/why you wrote each *line of code*, just each *section of the program*).

*15%* Justification of your solution. Marks awarded based on how clearly you explain all the factors that went into your decisions on how to solve the problem.

*40%* Working code. Full marks awarded if the TA can copy+paste your code into the simulator and it correctly implements the solution. For full marks, you must use interrupts — not polling — to read from the push buttons, as described above. *Your code must work without requiring any modifications to your code or the default settings of the simulator!*.

*10%* Writing quality. Marks awarded based on the extent to which your report is written coherently, in paragraph form, with complete sentences. Correct spelling and grammar are also included. Marks may also be deducted if the report is exhaustively long.

## *Resources & Lab Session*

A reference document for the necessary code in the ARMv7 language is available on OWL. The lesson material will also help explain some of these concepts. I will probably take some time in one of the lectures to provide additional tutorial support for this lab, this will be scheduled at some point.

- For this lab, you have the option of writing your program in C instead of ARMv7 Assembly, if you wish — as long as it will run in the simulator! (At the top of the main code window in the simulator there is a drop-down menu for selecting Assembly or C.) However I will generally not provide any trouble-shooting support if your code in C.
- A sample program (in both Assembly and C) that uses push button interrupts is available on OWL. You are free to copy this code for your project.