OBJECTIVES

In this lab you will write your first program for the Atmel processor. You will gain practice in programming the processor, understanding how to simulate your program **before** programming the board, and how to debug/emulate your program **after** programming the board.

REQUIRED MATERIALS

- As stated in the *Lab Rules and Policies*, submit your entire pre-lab report and **YOUR** asm file(s) through Canvas.
- As stated in the *Lab Rules and Policies*, print parts 13a) through 13d) of the pre-lab report for submission to your TA as you enter lab.
- Read/save the following:
  - *Create Simulate Emulate on Atmel* tutorial
  - *AVR instruction set (doc 0856)*
  - *Assembly Language Conversion: GCPU to Atmel Assembly*
- EEL 3744 Board kit and tools

PRELAB REQUIREMENTS

You **must** make a flowchart or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code. The flowchart or pseudo-code for parts B and C should be included (as stated in the *Lab Rules and Policies*, part 13g) in your pre-lab report.

PART A. ATMEL TUTORIAL

Go through the *Create Simulate Emulate on Atmel* tutorial found on the website. Obtain a screen shot on your laptop of the results of the last step (the emulation) that **shows your name** in big letters on the same screen. To create a screen shot in Windows, press Ctrl-PrtScn (i.e., select Ctrl and PrtScn at the same time). Save this screen shot in Appendix (section i) of your pre-lab report (as stated in the *Lab Rules and Policies*, part 13).

PART B. DEBUG A SIMULATED ASM PROJECT

Write a program (use filename **`lab1.asm`**) **using Atmel assembly language,** to filter data from an array in memory. For each byte in the data table, if the byte matches the filter condition, XOR the byte with **0x37** and the write this new data to the specified memory locations. Include this program in your pre-lab report (as stated in the *Lab Rules and Policies*, part 13h). Your program will assume that the data is already placed in program memory prior to execution, i.e., you will use assembler directives like ".DB" to put the constant values into memory. See the following webpage for more information on using assembler directives:

http://www.avr-tutorials.com/assembly/avr-assembler-directives

Section 4.5 of the below:
http://mil.ufl.edu/3744/docs/XMEGA/doc1022_Assembler_Directives.pdf

The array will be in hexadecimal and will contain the data in Table 1. ASCII is a 7-bit coded version of numbers, letters and symbols. An ASCII table can be found at www.asciitable.com. The most significant bit (bit 7) of each ASCII character is set to zero; because some of our data has this bit set as one, those bytes are not ASCII characters. You should therefore use the hexadecimal values to declare the data table in program memory.

Table 1: Memory Array

Byte Address (Hex)	Data (ASCII)	Data (Hex)
3744	??	0xA4
3745	p	0x70
3746	??	0x81
3747	X	0x58
3748	X	0x58
3749	S	0x53
374A	??	0x96
374B	??	0x17
374C	]	0x5D
374B	??	0xEE
374D	X	0x58
374E	??	0xF1
374F	??	0x83
3750	??	0xDB
3751	U	0x55
3752	??	0x99
3753	??	0x16
3754	??	0xC2
3755	??	0xD7
3756	??	0xF5
3757	NUL	0

Note that the end of the array is indicated by the "NUL" character, 0x00. Your program should filter the data in the array shown in Table 1 such that **only** characters with a hex value **less than** 0x80 **AND greater than or equal to** 0x16 are copied. XOR the data being copied with 0x37. Your program should end after it detects the "NUL" character. The data in the original array should **not** be corrupted. The new table should end by adding the NUL character.

The new table (the filtered data) should be placed in consecutive memory locations starting at address 0x2016 in **data memory**. You must use a pointer to accomplish this task.

When you write your program, be sure to make your code modular and use re-locatable, i.e., make it easy to change the location of both your input and output tables and use assembler directives appropriately. Use ".DSEG" for the output table. Try to make it easy to change the end of table value (NUL) and the filter values (0x16 and 0x80), i.e., use assembler directives for these two values.

Test your program using the Atmel Studio Simulator. Verify that the program works. Take a screen shot of a memory view window that shows the filtered values located at the correct addresses. Make sure the screenshot **displays your name** in big letters on the same screen. Save this screen shot in the Appendix (section 13i) of your pre-lab report (as stated in the *Lab Rules and Policies*, part 13).

PART C. DEBUG AN EMMULATED ASM PROJECT

In this part you will program your UF Board with the program you wrote in Part B. Debugging on the actual hardware is called **emulation**. Whenever possible, you should simulate your design first, before emulating; this eliminates any chance of mistaking hardware bugs for software bugs. Load the program you created for Part B onto your board and verify that it functions properly. Again obtain a screenshot of the new table in memory. Make sure the screenshot **displays your name** in big letters on the same screen. Save this screen shot in the Appendix of your pre-lab report (as stated in the *Lab Rules and Policies*, part 13i).

PRELAB PROCEDURE

1. It is required that you make a flowchart or write pseudo-code **before** writing **any** program in this course. This will help you formulate a plan of attack for the code.
2. Create your program for parts B and C (in a file on your computer) using the Atmel assembly language instruction set. Include this in your pre-lab report (as stated in the *Lab Rules and Policies*, part 13h).
3. Bring the required printed parts of your pre-lab report to turn in to your TA (as stated in the *Lab Rules and Policies*, part 13a-13d).

Note: All pre-lab requirements **MUST** be accomplished **PRIOR** to coming to your lab.

PRELAB QUESTIONS

1. What instruction can be used to read from program memory (flash)? Can you use any registers with this instruction?
2. What is the difference between program and data memory? See section 7 in the ATxmega128A1U manual
3. When using RAM (not EEPROM), what memory locations can be utilized for the .DSEG? Why? What .DSEG did you use in this lab and why?

LAB PROCEDURE

Simulation Demo:

Demonstrate (to your TA) that your program from part B creates the correct table. The TA will ask you to change the data values and/or filtering condition and then re-simulate/emulate your program. Your TA will also ask you to single step through your program and use breakpoints. Be prepared to answer questions about your program.

Downloading program and Emulate:

Demonstrate that you are able to emulate your program from part B on your UF board.

Time permitting:

If you have time available, read lab 2 and start building and testing your LED (output) and switch (input) circuits.