Lab 1: Programming the Atmel XMEGA

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*, email your entire pre-lab report and **YOUR** asm file(s).
- As stated in the *Lab Rules and Policies*, print parts 6a) through 6d) of the pre-lab report.
- Read/save the following:
  - *Create Simulate Emulate on Atmel* tutorial
  - *AVR instruction set (doc 0856)*
  - *Assembly Language Conversion: GCPU to Atmel Assembly*
- EEL 4744 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 6g) 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 6).

PART B. DEBUG A SIMULATED ASM PROJECT

Write a program, **using Atmel assembly language,** to filter data from an array in memory and copy that filtered data to specified memory locations. Include this program in part h) of your pre-lab report (as stated in the *Lab Rules and Policies*, part 6). 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 examples using assembler directives:

http://support.atmel.no/knowledgebase/avrstudiohelp/mergedprojects/avrasm/html/directives.html

The array will be in ASCII code and will contain the data in Table 1. An ASCII table can be found at www.asciitable.com. ASCII is a 7-bit coded version of numbers, letters and symbols. The most significant bit (bit 7) is set to zero in our example and in the given website.

Table 1: Memory Array

Address	Data (ASCII)	Data (Hex)
0x0500	A	0x41
0x0501	u	0x75
0x0502	~	0x7e
0x0503	(space)	0x20
0x0504	!	0x21
0x0505	4	0x34
0x0506	P	0x50
0x0507	_	0x5F
0x0508	7	0x37
0x0509	4	0x34
0x050A	i	0x69
0x050B	4	0x34
0x050C	I	0x49
0x050D	G	0x47
0x050E	C	0x43
0x050F	s	0x73
0x0510	_	0x5F
0x0511	f	0x66
0x0512	O	0x4f
0x0513	L	0x4c
0x0514	(space)	0x20
0x0515	u	0x75
0x0516	z	0x7a
0x0517	%	0x25
0x0518	!	0x21
0x0519	n	0x6e
0x051A	NUL	0x00

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** 0x76 **AND greater than or equal to** 0x50 are copied. Your program should end after it copies the "NUL" character. The data in the original array should **not** be corrupted. The new table should end with the NUL character (which should be copied).

The new table (the filtered data) should be placed in consecutive memory locations starting at address 0x3700 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 value (0x50 and 0x76), i.e., use assembler directives for these two values.

Test your program using the AVR 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 i) of your pre-lab report (as stated in the *Lab Rules and Policies*, part 6).

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 6i).

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 6h).
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 6a-6d).

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

PRELAB QUESTIONS

1. What is the difference between program and data memory? See section 7 in the ATxmega128A1U manual
2. What memory location should the .DSEG section start? Why?
3. What registers can we use to read from program memory (flash)?

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.