EEL4742

For the purposes of this homework, you will be examining the AVR family of microcontrollers architecture. Specifically, we will be looking at the megaAVR and tinyAVR architectures. Development of AVR started in 1996, and currently serves as the heart of the popular Arduino platform. We briefly mentioned AVR during class. Recall that AVR is a modified Harvard architecture, that is, it provides program and data spaces in separate address busses while also providing a mechanism to transfer data between the two. In this homework, you will be investigating the basis of the architecture and writing a few assembly programs for it. We recommend looking for the *AVR Instruction Set Manual* as well as other documentation you may find in Atmel's website.

Q.1 Describe the register file of AVR microcontrollers: How many registers are in the register file? How wide are they? Are there any special purpose registers like in MSP430? Where are the stack pointer and status register located?

Q.2 Describe the calling convention utilized by Atmel in their compiler.

Q.3 Utilize your findings from Q.1 and Q.2 to answer the following question. We desire to add two 16bit numbers, 1455 and 3437. Write a small AVR assembly language program to add these two numbers.

Q.4 What are the `X`, `Y`, and `Z` registers? Write a small assembly program that loads constants into these registers.

Q.5 Recall from the lectures that for convenience we store variables with permanent store (local variables declared as static and global variables) in the `.data` or `.bss` sections of memory. The `.data` section contains variables with permanent store that have been explicitly initialized, whereas the `.bss` section contains variables with permanent store that have not been explicitly initialized (these are implicitly initialized to 0). Recall that because we do not have a program loader, our program is responsible for performing this initialization.
Fortunately, when the final binary is being linked, the linker script exports the symbols shown in Table 1. Utilize these symbol names to write two assembly routines, `__initialize_data` and

Name	**Description**
`__data_load_start`	Location in *program memory* where the contents that initializes the `.data` section is stored.
`__data_load_end`	Location in *program memory* where the contents that initializes the `.data` section ends.
`__data_start`	Location in *data memory* where the contents of the `.data` section should reside.
`__data_end`	Location in *data memory* where the contents of the `.data` section should end.
`__bss_start`	Location in *data memory* where the contents of the `.bss` section should start.
`__bss_end`	Location in *data memory* where the contents of the `.bss` section should end.

Table 1: Symbols exported by the linker script in AVR.

`__initialize_bss` that initializes the `.data` and `.bss` sections of data memory, respectively on the AVR architecture.