Aaron Vaughan

10/24/2019

ECE 375

Pre-Lab

1. What is the stack pointer? How is the stack pointer used, and how do you initialize it? Provide pseudocode (not actual assembly code) that illustrates how to initialize the stack pointer.

a. The stack pointer points to the address above the top of stack so that when you want to push an item onto the stack the stack pointer is currently pointing to the address at which the data will be pushed into. When data is pushed the stack pointer is decremented and when a data is popped the stack pointer is incremented.

b. Pseudo-Code:

SPL gets low byte of RAMEND

SPH gets high byte of RAMEND

2. What does the AVR instruction LPM do, and how do you use it? Provide pseudocode (not actual assembly code) that shows how to setup and use the LPM instruction.

a. LPM is the mnemonic for load program. It is most useful for accessing constants stored in program memory. LPM is used by initializing the Z-pointer to a

value one byte at a time. Then you can load a constant from program memory pointed to by Z into a given register.

```
High bit of Z gets a value      ; initialize high bit
Low bit of Z gets a value       ; initialize low bit
R16 gets Z                      ; load program memory
```

3. Take a look at the definition file m128def.inc (This file can be found in the Solution Explorer → Dependencies folder in Atmel Studio, assuming you have an Assembler project open and you have already built an assembly program that includes this definition file. Two good examples of such a project would be your Lab 1 and Lab 3 projects.) What is contained within this definition file? What are some of the benefits of using a definition file like this? Please be specific, and give a couple examples if possible.

a) The file is full of .equ designations. This helps the programmer by allowing the use of mnemonics to describe ports, timers, registers and more. It includes a few .def statements that set up the low and high bytes of the X, Y, and Z pointers. The mnemonics could be standard over the spectrum of avr boards so that a programmer does not need to specifically memorize where each byte of the X, Y, and Z pointers are. In the first and second labs we were able to refer to the input values by their PINx names rather than some specific memory mapped address.