ECE 375 PRELAB 4

Lab Time: Wednesday 10-12

Tu Lam

QUESTIONS

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.*

Stack pointer is used to point to the top of the stack. The stack pointer is used to point to the address in the stack that hold the information/value about the active subroutines of a program, i.e… Below is a pseudocode on how to initialize the stack pointer:

```
Ldi  mpr, high(RAMEND)

Out  SPH, mpr

Ldi  mpr, low(RAMEND)

Out  SPL, mpr
```

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.*

LPM or Load program memory is used to access either 8-bit or 16-bit constants stored in the Program Memory. Below is a pseudocode for it:

```
LPM  Rd, Z

LPM  Rd, Z+
```

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.*

The m128def.inc is which contains lots of .EQU and a few .DEF directives, as well as other useful information, such as the last address in Data Memory (RAMEND). The benefit of this definition file is that contains addresses and values for common I/O registers and special registers within a specific processor which is good when it comes using the AVR. An example is every Atmel AVR processor contains SREG, but its location in memory may not be the same across different processors. Thus, this can be used to define common names for I/O registers, such as SREG and SPH, so that programmers do not have to memorize or look up each I/O register or processor specific registers.

REFERENCE

Computer Organization and Assembly Language Programming: Embedded Systems Perspective by Ben Lee