Rhea Mae Edwards
ECE 375

LAB 4 PRELAB

1. The *stack pointer* is a register that always points to the top of the stack, where the stack is mainly used to store data temporary.

   You use the stack pointer with various commands; common ones being PUSH and POP, where the PUSH command would decreases the stack pointer, and the POP command would increase the stack pointer, since the stack is implemented where memory locations grows from higher to lower memory locations. The AVR stack pointer is implemented as two 8-bit special function registers- stack pointer high register (SPH) and stack pointer low register (SPL).

   To initialize the stack pointer, you would include a definition file at the beginning of a program to utilize register naming schemes (represented below), and then perform the following:

   1. Include the definition file in the program
   2. Read in the low byte of the end SRAM address
   3. Write the byte to SPL
   4. Read in the high byte of the end SRAM address
   5. Write the byte to SPH

2. The AVR instruction *LPM* (Load Program Memory) loads one byte pointed to by the z-register into the destination register Rd.

   You would use the LPM instruction to achieve a 100% space effective constant initialization or constant data fetch.

   To setup and use the LPM instruction, you would perform the following:

   1. Initialize the z-pointer
   2. Load a constant from the program memory, so then the memory is pointed to by z (R31:R30)
   3. Z-pointer register then can either be left unchanged or be incremented by the operation

3. The *m128def.inc definition file* contains all the I/O register names and I/O register bit names that can be used within an assembly program, in addition to the six register forming the three data pointers X, Y, and Z being assigned the names XL – ZH, where the highest RAM address for Internal SRAM is also defined within this definition file.

   The benefits that come with using a definition file like this involves the utilization of the registers' naming schemes, since it contains addresses and values for common I/O registers and special registers within a specific chipset. The idea is that the programmer does not have to look up or memorize the address for each of the I/O register or chip specific registers, and that also the same code can be used for different chipset by just including the proper definition file. For example, every ATMEL AVR chipset contains an SREG, but not every chipset has the SREG in the same memory location, which is now where the definition file comes in handy.