Goals:
Understand the basic functions the compiling and debugging in AVR Studio 5.
Understand the basic functions of the AVRISP mkII.

Equipment Usage:
For this lab the following equipment will be used:

- Atmel Studio 6
- AVR RISP mkII
- ATmega328P

Background:
User Guide for AVRISP mkII:

http://people.ece.cornell.edu/land/courses/ece4760/AtmelStuff/avrispmkii_ug.pdf

or

http://www.atmel.no/webdoc/index.html (select user guide under AVR ISP mkII section)

Connecting to the AVRISP mkII:

For the majority of these labs we will be using the AVRISP mkII. The figure below displays the required connections needed for the ISP.

![Diagram of AVRISP mkII connections]

Using the ATmega328P, the same ports must be attached to the ISP. Failure to connect to the proper ports will result in an error displayed when the AVRISP mkII attempts to read the attached device. You will have to refer to the datasheet of each IC to see the location of each port (SCK, MISO, MOSI) since they will be different for each chip.
Assembly Code:

Assembly is a low-level programming language that allows users to generate expressions that are equivalent to machine language instructions. Because assembly is essentially machine language, it is often used instead of high-level programming languages (such as C, C++, Java, etc.) when programming devices such as microcontrollers and microprocessors. Simple instructions such as \( c = a + b \) are much longer in assembly code. The benefit of this is that assembly language provides a programmer more control over the instructions.

Basic ASM Functions

<table>
<thead>
<tr>
<th>Mnemonics</th>
<th>Operands</th>
<th>Description</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD</td>
<td>Rd, Rr</td>
<td>Add two Registers</td>
<td>Rd ← Rd + Rr</td>
</tr>
<tr>
<td>SUB</td>
<td>Rd, Rr</td>
<td>Subtract two Registers</td>
<td>Rd ← Rd - Rr</td>
</tr>
<tr>
<td>AND</td>
<td>Rd, Rr</td>
<td>Logical AND Registers</td>
<td>Rd ← Rd &amp; Rr</td>
</tr>
<tr>
<td>OR</td>
<td>Rd, Rr</td>
<td>Logical OR Registers</td>
<td>Rd ← Rd | Rr</td>
</tr>
<tr>
<td>NEG</td>
<td>Rd</td>
<td>Two's Complement</td>
<td>Rd ← 0x00 − Rd</td>
</tr>
<tr>
<td>INC</td>
<td>Rd</td>
<td>Increment</td>
<td>Rd ← Rd + 1</td>
</tr>
<tr>
<td>DEC</td>
<td>Rd</td>
<td>Decrement</td>
<td>Rd ← Rd − 1</td>
</tr>
<tr>
<td>RJMP</td>
<td>k</td>
<td>Relative Jump</td>
<td>PC ← PC + k + 1</td>
</tr>
<tr>
<td>RCALL</td>
<td>k</td>
<td>Relative Subroutine Call</td>
<td>PC ← PC + k + 1</td>
</tr>
<tr>
<td>RET</td>
<td></td>
<td>Subroutine Return</td>
<td>PC ← STACK</td>
</tr>
<tr>
<td>BREQ</td>
<td>k</td>
<td>Branch if Equal</td>
<td>if (Z = 1) then PC ← PC + k + 1</td>
</tr>
<tr>
<td>BRNE</td>
<td>k</td>
<td>Branch if Not Equal</td>
<td>if (Z = 0) then PC ← PC + k + 1</td>
</tr>
<tr>
<td>MOV</td>
<td>Rd, Rr</td>
<td>Move Between Registers</td>
<td>Rd ← Rr</td>
</tr>
<tr>
<td>LDI</td>
<td>Rd, K</td>
<td>Load Immediate</td>
<td>Rd ← K</td>
</tr>
<tr>
<td>ST</td>
<td>X, Rr</td>
<td>Store Indirect</td>
<td>(X) ← Rr</td>
</tr>
<tr>
<td>IN</td>
<td>Rd, P</td>
<td>In Port</td>
<td>Rd ← P</td>
</tr>
<tr>
<td>OUT</td>
<td>P, Rr</td>
<td>Out Port</td>
<td>P ← Rr</td>
</tr>
<tr>
<td>PUSH</td>
<td>Rr</td>
<td>Push Register on Stack</td>
<td>STACK ← Rr</td>
</tr>
<tr>
<td>POP</td>
<td>Rd</td>
<td>Pop Register from Stack</td>
<td>Rd ← STACK</td>
</tr>
</tbody>
</table>

Prelab:

Watch the videos explaining how to create and simulate programs using Atmel Studio


Lab Experiments:

**Setup:** To setup the circuit for this lab you should program your chip first while it is still connected to its development board. Once the IC is programmed, remove it and place it on a breadboard and set all VCC (also AVCC for 328P) connections to 5V and all GND connections to 0V.
**Experiment 1:** Program the following assembly code using Atmel Studio 6 into your Atmega 328P. The code should light up an LED connected to pin (PB.4)

```
.org 0
SBI DDRB,4 ;set PB4 as an output
LDI R17,16 ;value is 16 to light up bit 4
OUT PORTB,R17 ;sends value of R17 to corresponding bit
```

**Experiment 2:**
Program the following C code using AVR studio with your Atmega 328P
```
#include <avr/io.h>

int main ()
{
    DDRB = 0xFF; //set all of portB to output
    while (1)
    {
        PORTB = 0x88; //set PB7 and PB3 high
        return 0;
    }
}
```

**Experiment 3:**
Modify the given assembly code and program the ATmega328P to light up the LEDs connected to pin PB5 and PB3 simultaneously. Modify the given C code for the ATmega328P light LEDs connected to pins PB6, PB4, PB2 simultaneously.
Post-Lab Deliverables:

1) Copy of your modified code
2) Questions:
   a. What does the red stripe of the ISP cable indicate?
   b. List and explain the different color codes of the AVRISP mkII
   c. At what frequency should the programmer be set at to read/write to the device?
   d. Write code in assembly that will perform the following equation: 100+53-27
   e. Explain what is happening on each line of the following code. If you were to execute this code what would be the final decimal value stored in R20?

```
LDI R20, 0x75
LDI R21, 0x05
LDI R22, 0x24
ADD R20, R22
SUB R22, R21
ADD R20, R21
SUB R22, R20
ADD R20, R22
MOV R20, R21
RJMP DONE
ADD R21, R20
SUB R21, R22
DONE:
SUB R20, R21
END: RJMP END
```