10/16 Lecture Notes

Take a look at the cheat sheet! Try to remember some of the more basic operations.
Pop Quiz
1. Load the value 114 into register r1, load the value -13 into r2, add them and place the result in r3.

We cannot directly do this:
```
ldi r1, 114
```
because ldi only accepts r16-r31.

Let's take a look at
```
ld r1, X
```

What does it do? It loads the value stored in the RAM address indicated by the values in r26 and r27 using the formula 256 * r27 + r26. Since we haven't loaded anything into r26 and into r27, this calculates to address 0. Thus, this will load whatever is in RAM address 0 into r1.

```
;load
ldi r17, 114
mov r1, r17
ldi r17, -13
mov r2, r17

; add
ldi r17, 0
mov r3, r17
add r3, r1
add r3, r2
```

or

```
mov r3, r1
add r3, r2
```

or
```
add r1, r2
mov r3, r1
```

Be careful when loading/modifying registers.

Does it matter that r17 is DDRD?

The general purpose registers are separate from the IO registers! IO registers can only be manipulated by the “out/in” instructions. They are used for different purposes and r17 has nothing to do with I/O register 17.

2. Output the register r3 using an out instruction so the output appears on the simulators IO.

Consider
```
out r3
```

and

```
out 18, r3
```

What is the difference? The first one is syntactically illegal. The first argument must be a I/O register, the second argument is a general-purpose register.

Before that, though, you need to confirm that PORTD is set to all output. So, you need to do:

```
ldi r16, 255
out 17, r17
out 18, r3
```

It takes 255, which is all 1's, and loads it into r16. Then, 255 is placed into “status register” IO register 17. 17 is a configuration register that configures all the parts of IO 18 as output. In this course, we will set 17 to either be exclusively output (all 1's) or exclusively input (all 0's).

3. Initialize sp (stack pointer) to the value 2146
2146 = 8*256 + 98

Why should we initialize the stack pointer? It is used to store values in memory for function calls. In more advanced languages, the compiler will handle this stuff for you. The location of the stack pointer indicates how much memory you want to allocate to the stack.

The main thing to remember is that IO registers 61 and 62 are “special” registers specific to the stack and together they form a calculation that indicates the location of the stack pointer.

```
ldi r31, 8
out 62, r31
ldi r31, 98
out 61, r31
```

For the general purpose registers, you are free to choose whichever you want.

Do you need to clear r31 after you use it (ie. Set it to all zeros)? Answer: no. Just keep track of which value is held in which register at a given point in time.

Can we do 61 first and then 62? Yes! The stack pointer can be treated as a single logical entity so it doesn't matter what order we initialize it in. However, it is fixed which one is going to be multiplied by 256 (it's 62). sp=[iore62:iore61].

4. Write assembly language code to count the number of 1s in a number. You can assume the number has been loaded into r30.

This will be left for Monday to be done in-class.