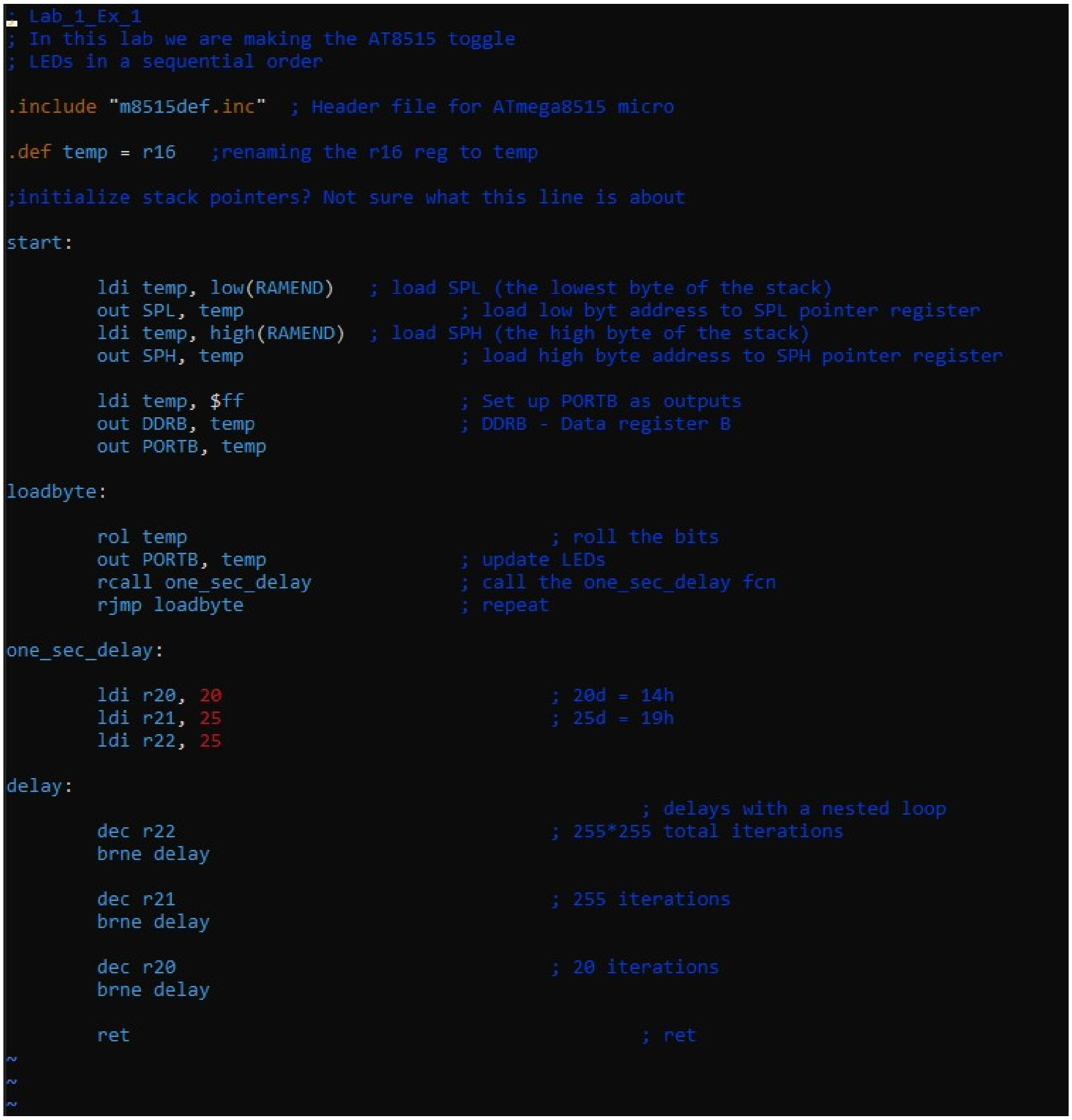
Lab 1
Jacob Hillebrand
CEE-345 Microprocessor System Design

Part 1

For this part, we toggled the LEDs on and off on the STK-600 board with a blinking pattern. This part was done in Assembly.
The code loaded the board's registers, set up the proper ports, output the proper values to said ports, created an LED pattern, created a delay, then iterated through the entire process repeatedly.

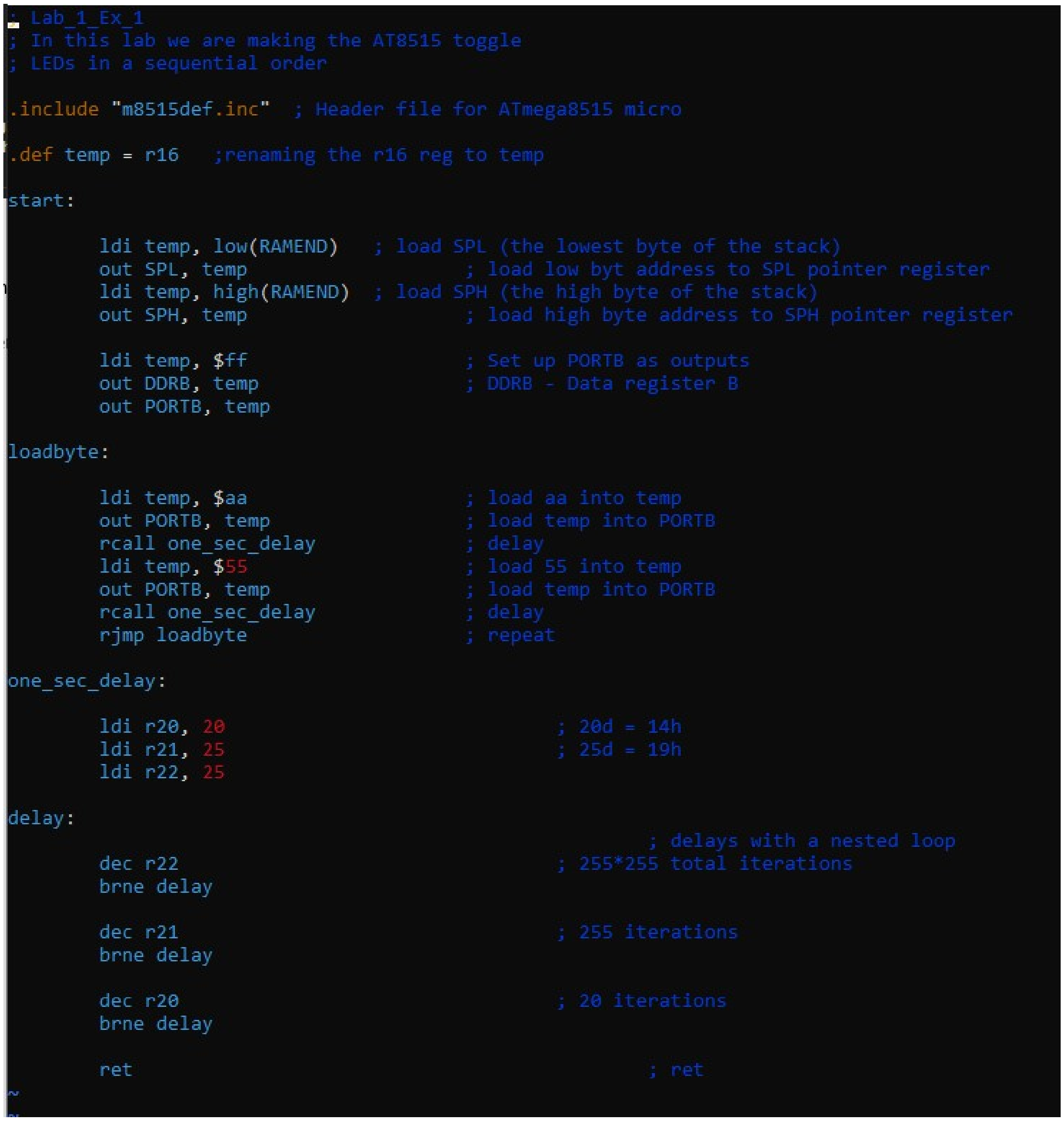


```
; Lab_1_Ex_1
; In this lab we are making the AT8515 toggle
; LEDs in a sequential order

.include "m8515def.inc"  ; Header file for ATmega8515 micro

.def temp = r16   ;renaming the r16 reg to temp

;initialize stack pointers? Not sure what this line is about

start:

        ldi temp, low(RAMEND)   ; load SPL (the lowest byte of the stack)
        out SPL, temp                   ; load low byt address to SPL pointer register
        ldi temp, high(RAMEND)  ; load SPH (the high byte of the stack)
        out SPH, temp                   ; load high byte address to SPH pointer register

        ldi temp, $ff                   ; Set up PORTB as outputs
        out DDRB, temp                  ; DDRB - Data register B
        out PORTB, temp

loadbyte:

        rol temp                                ; roll the bits
        out PORTB, temp                 ; update LEDs
        rcall one_sec_delay             ; call the one_sec_delay fcn
        rjmp loadbyte                   ; repeat

one_sec_delay:

        ldi r20, 20                             ; 20d = 14h
        ldi r21, 25                             ; 25d = 19h
        ldi r22, 25

delay:
                                                        ; delays with a nested loop
        dec r22                                 ; 255*255 total iterations
        brne delay

        dec r21                                 ; 255 iterations
        brne delay

        dec r20                                 ; 20 iterations
        brne delay

        ret                                             ; ret
~
~
~
```

Figure 1: Code from Part 1

Part 2

For this part, we modified the code from Part 1 such that it generated an LED pattern where every other LED toggled on/off at the same time. This part was also written in Assembly Again, this code loaded the board's registers, created two different LED patterns, created the proper delays for both patterns, set up the ports, and iterated through the entire process.

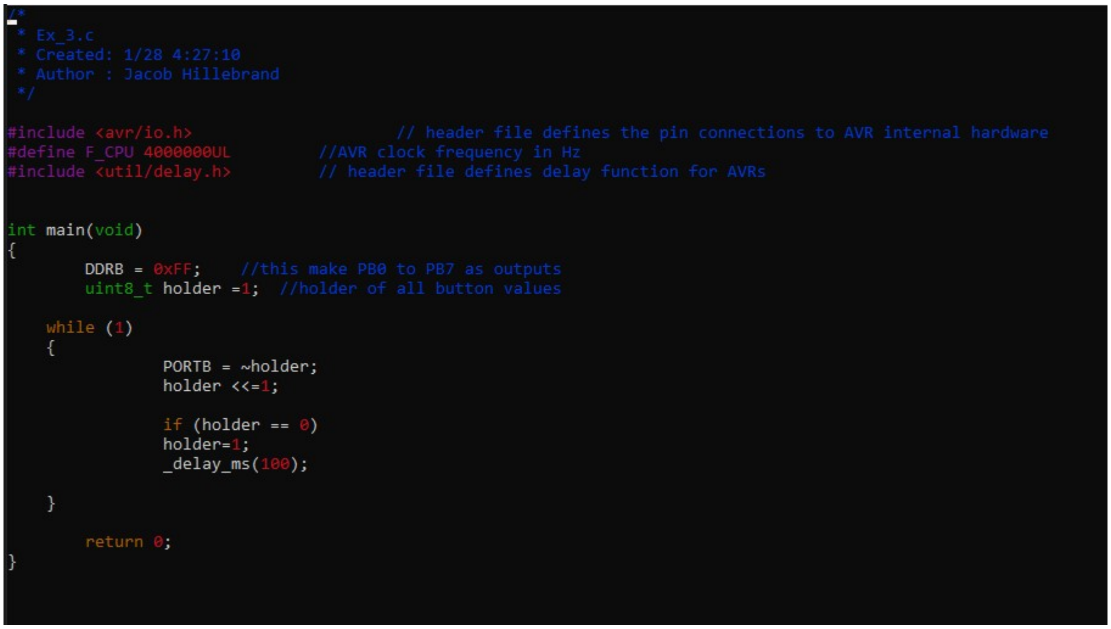


```
; Lab_1_Ex_1
; In this lab we are making the AT8515 toggle
; LEDs in a sequential order

.include "m8515def.inc"  ; Header file for ATmega8515 micro

.def temp = r16   ;renaming the r16 reg to temp

start:

        ldi temp, low(RAMEND)   ; load SPL (the lowest byte of the stack)
        out SPL, temp                   ; load low byt address to SPL pointer register
        ldi temp, high(RAMEND)  ; load SPH (the high byte of the stack)
        out SPH, temp                   ; load high byte address to SPH pointer register

        ldi temp, $ff                   ; Set up PORTB as outputs
        out DDRB, temp                  ; DDRB - Data register B
        out PORTB, temp

loadbyte:

        ldi temp, $aa                   ; load aa into temp
        out PORTB, temp                 ; load temp into PORTB
        rcall one_sec_delay             ; delay
        ldi temp, $55                   ; load 55 into temp
        out PORTB, temp                 ; load temp into PORTB
        rcall one_sec_delay             ; delay
        rjmp loadbyte                   ; repeat

one_sec_delay:

        ldi r20, 20                             ; 20d = 14h
        ldi r21, 25                             ; 25d = 19h
        ldi r22, 25

delay:
                                                        ; delays with a nested loop
        dec r22                                 ; 255*255 total iterations
        brne delay

        dec r21                                 ; 255 iterations
        brne delay

        dec r20                                 ; 20 iterations
        brne delay

        ret                                             ; ret
~
~
```

Figure 2: Code from Part 2

Part 3

For the final part of this Lab, we rewrote Part 1 in C instead of Assembly. The code set the clock speed for the controller, created the LED pattern, created a time delay, and cycled through this sequence 8 times to show the moving LED. Then, this entire process was iterated repeatedly.



```
/*
 * Ex_3.c
 * Created: 1/28 4:27:10
 * Author : Jacob Hillebrand
 */

#include <avr/io.h>                     // header file defines the pin connections to AVR internal hardware
#define F_CPU 4000000UL         //AVR clock frequency in Hz
#include <util/delay.h>         // header file defines delay function for AVRs


int main(void)
{
        DDRB = 0xFF;    //this make PB0 to PB7 as outputs
        uint8_t holder =1;  //holder of all button values

    while (1)
    {
                PORTB = ~holder;
                holder <<=1;

                if (holder == 0)
                holder=1;
                _delay_ms(100);

    }

        return 0;
}
```

Figure 3: Code from Part 3